

**FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO**

# **Bio-Measurements Estimation and Support in Knee Recovery through Machine Learning**

**João Miguel Neves Bernardino**



**Mestrado Integrado em Engenharia Informática e Computação**

**Supervisor: Hugo Sereno Ferreira, PhD**

**Second Supervisor: Luís Filipe Teixeira, PhD**

**External Advisor: Fábio Rocha, Physical Therapist**

**July 23, 2018**



# **Bio-Measurements Estimation and Support in Knee Recovery through Machine Learning**

**João Miguel Neves Bernardino**

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Prof. André Restivo

External Examiner: Prof. Pedro Ribeiro

Supervisor: Prof. Hugo Sereno Ferreira

July 23, 2018



# Abstract

The knee is the most complex joint of the human body, having multiple ligaments and tendons in its composition. As a consequence, many distinct injuries can occur, often requiring the patient to undergo intensive rehabilitation during several months. Regardless of the exact nature of the injury, treatment protocols usually contemplate some recurrent measurements and evaluations that provide valuable information for enhancing the rehabilitation process. Unfortunately, due to the high cost of some required equipment or the fact that some of these procedures are complex and time-consuming, they are often neglected.

A deep learning based solution for performing measurements of knee range of motion using convolutional neural networks is presented, supported by the generation of a synthetic dataset. A 3D human-body model was used to generate realistic and varied images, perpendicular to the plane defined by the lower limb, simulating a patient lying prone on a treatment table with the leg flexed at arbitrary angles. Such images were labeled, being registered the coordinates of three key points for the calculation of the flexion angle: the centers of the thigh, the knee and the lower leg. This data was used to train convolutional neural networks for performing regression and predicting these six coordinates. Transfer learning was used with the VGG16 and InceptionV3 models pre-trained on the ImageNet dataset. An additional custom model was trained from scratch. These networks were tested under various conditions using different combinations of image augmentation techniques on the training sets. Out of the three architectures, Eva achieved the best results, closely followed by InceptionV3. VGG16's results were unsatisfactory. Posterior testing was performed using a few real images to test how well the network generalized, and the results were also favorable.

Future work would involve further improvement of the solution and its integration in a mobile application which could be used by physiatrists and physical therapists, providing a superior level of usability in comparison to current methods used for this type of measurement, ultimately optimizing the rehabilitation process.

**CCS Concepts:** Computing methodologies → Artificial intelligence → Computer vision → Computer vision problems → Object recognition

**Keywords:** knee, rehabilitation, measurements, goniometry, range of motion, physical therapy, computer vision, machine learning, deep learning, convolutional neural networks, smartphone, camera



*“Whether you think you can or think you can’t,  
you’re right.”*

Henry Ford





# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Context . . . . .	1
1.2	Motivation . . . . .	2
1.3	Goals . . . . .	3
1.4	Report Structure . . . . .	3
<b>2</b>	<b>State of the Art</b>	<b>5</b>
2.1	Measurements in Knee Rehabilitation . . . . .	5
2.1.1	Goniometry . . . . .	5
2.2	Smartphone-based Solutions . . . . .	7
2.2.1	Inertial-based goniometry . . . . .	7
2.2.2	Photography-based goniometry . . . . .	8
2.3	Conclusion . . . . .	9
<b>3</b>	<b>Background</b>	<b>11</b>
3.1	Image Recognition . . . . .	11
3.1.1	Computer vision . . . . .	11
3.1.2	Machine learning . . . . .	12
3.1.3	Deep learning . . . . .	12
3.1.4	Convolutional neural networks . . . . .	12
3.2	Transfer Learning . . . . .	14
3.3	Deep Learning with Synthetic Data . . . . .	14
3.4	Supporting Frameworks and Software . . . . .	14
3.4.1	MakeHuman . . . . .	14
3.4.2	Blender . . . . .	15
3.4.3	Keras . . . . .	16
3.4.4	CoreML . . . . .	16
3.5	Conclusion . . . . .	16
<b>4</b>	<b>Problem Statement</b>	<b>17</b>
4.1	Requirements . . . . .	17
4.2	Solution Proposal . . . . .	18
4.3	Evaluation . . . . .	18
4.4	Conclusion . . . . .	18
<b>5</b>	<b>Synthetic Dataset Generation</b>	<b>21</b>
5.1	3D Model Creation . . . . .	21
5.2	Manipulation and Posing . . . . .	22

## CONTENTS

5.3	Automated Data Generation . . . . .	23
5.4	Conclusion . . . . .	25
<b>6</b>	<b>Image Recognition</b>	<b>27</b>
6.1	Supporting Libraries and Resources . . . . .	27
6.2	Image Augmentation . . . . .	28
6.3	CNN Architectures . . . . .	30
6.3.1	InceptionV3 . . . . .	31
6.3.2	VGG16 . . . . .	32
6.3.3	Eva . . . . .	32
6.4	Training . . . . .	33
6.4.1	Test #1: No augmentation . . . . .	35
6.4.2	Test #2: Horizontal flips . . . . .	36
6.4.3	Test #3: Translations . . . . .	37
6.4.4	Test #4: Background images . . . . .	38
6.4.5	Test #5: Rotations . . . . .	39
6.4.6	Test #6: Flips and translations . . . . .	40
6.4.7	Test #7: Flips, translations and rotations . . . . .	41
6.4.8	Test #8: Flips, translations, rotations and backgrounds . . . . .	42
6.5	Conclusion . . . . .	43
<b>7</b>	<b>Conclusions</b>	<b>45</b>
7.1	Difficulties . . . . .	46
7.2	Future Work . . . . .	46
7.2.1	Diversity of the dataset . . . . .	46
7.2.2	Different CNN architectures . . . . .	46
7.2.3	Predicting vectors instead of points . . . . .	47
7.2.4	Predicting the flexion angle directly . . . . .	47
7.2.5	Mobile application . . . . .	47
7.2.6	On-site validation . . . . .	47
7.2.7	Augmented reality . . . . .	47
7.2.8	Additional measurements . . . . .	48
7.3	Epilogue . . . . .	48
	<b>References</b>	<b>49</b>

# List of Figures

1.1	Anatomy of the human knee . . . . .	2
2.1	Measurement of joint angle in x-ray . . . . .	6
2.2	Universal goniometer . . . . .	6
2.3	Digital protractor . . . . .	7
2.4	Knee goniometry using a gyroscope-based application . . . . .	8
2.5	Knee ROM measured with DrGoniometer . . . . .	9
3.1	Typical structure of a CNN . . . . .	13
3.2	MakeHuman's main interface . . . . .	15
3.3	Blender's main interface . . . . .	15
5.1	MakeHuman 3D models . . . . .	22
5.2	Different knee flexion angles in Blender . . . . .	23
5.3	Examples of different skin textures used . . . . .	23
5.4	Examples of synthesized data . . . . .	24
6.1	Examples of the transformations applied to the dataset . . . . .	28
6.2	Some of the photographs used as background . . . . .	29
6.3	Diagram representing the required CNN structure . . . . .	31
6.4	Samples from the validation sets . . . . .	34
6.5	Training curves for Test #1 . . . . .	35
6.6	Sample predictions after Test #1 . . . . .	35
6.7	Training curves for Test #2 . . . . .	36
6.8	Sample predictions after Test #2 . . . . .	36
6.9	Training curves for Test #3 . . . . .	37
6.10	Sample predictions after Test #3 . . . . .	37
6.11	Training curves for Test #4 . . . . .	38
6.12	Sample predictions after Test #4 . . . . .	38
6.13	Training curves for Test #5 . . . . .	39
6.14	Sample predictions after Test #5 . . . . .	39
6.15	Training curves for Test #6 . . . . .	40
6.16	Sample predictions after Test #6 . . . . .	40
6.17	Training curves for Test #7 . . . . .	41
6.18	Sample predictions after Test #7 . . . . .	41
6.19	Training curves for Test #8 . . . . .	42
6.20	Sample predictions after Test #8 . . . . .	42
6.21	InceptionV3 predictions on edited photographs (background removed) . . . . .	43
6.22	InceptionV3 predictions on real unedited photographs . . . . .	43

## LIST OF FIGURES

# List of Tables

5.1	Example of data stored in the CSV file . . . . .	24
6.1	InceptionV3 final fully-connected layers (modified) . . . . .	31
6.2	VGG16 final fully-connected layers (modified) . . . . .	32
6.3	Layers of the Eva CNN model . . . . .	33
6.4	Results of the 8 test scenarios. T and V columns represent the minimum loss values on the training and validation sets, respectively. Lower results are better. . . . .	43

## LIST OF TABLES

# Abbreviations

ACL	Anterior cruciate ligament
CNN	Convolutional neural network
CSA	Cross-sectional area
DL	Deep learning
IT	Information technology
ML	Machine learning
ROM	Range of motion
UG	Universal goniometer





# Chapter 1

## Introduction

---

<b>1.1</b>	<b>Context</b>	<b>1</b>
<b>1.2</b>	<b>Motivation</b>	<b>2</b>
<b>1.3</b>	<b>Goals</b>	<b>3</b>
<b>1.4</b>	<b>Report Structure</b>	<b>3</b>

---

The main purpose of this first chapter is to provide some insight on the problem that is in the origin of this project. Section [1.1](#) contains a brief description of its context. Sections [1.2](#) and [1.3](#) respectively explain the reasons for exploring a novel and different solution to the problem and the expected outcome of this work. Lastly, in section [1.4](#), the structure of this document is reviewed in detail.

### 1.1 Context

The knee is the largest and most anatomically complex joint in the human body [[GR03](#)]. It is formed between three bones and contains multiple tendons and ligaments, as seen on figure 1.1. Its complicated structure, combined with the great amount of stress caused by the forces it is subject to, makes it one of the most frequently injured joints, especially among athletes. A study published in 2006 revealed that, from 19.530 sports injuries documented over a 10-year period, close to 40% were related to the knee [[MSK06](#)].

The most common types of injury usually involve ruptures, either partial or complete, of some structure of the knee, such as cartilage, ligaments or tendons. In a great number of cases, especially the ones involving complete ruptures, surgical intervention might be necessary. Since the soft structures of the knee are very delicate, post-operative recovery time is usually long. For instance, an athlete who suffers an anterior cruciate ligament (ACL) full tear, which is one of the most

## Introduction

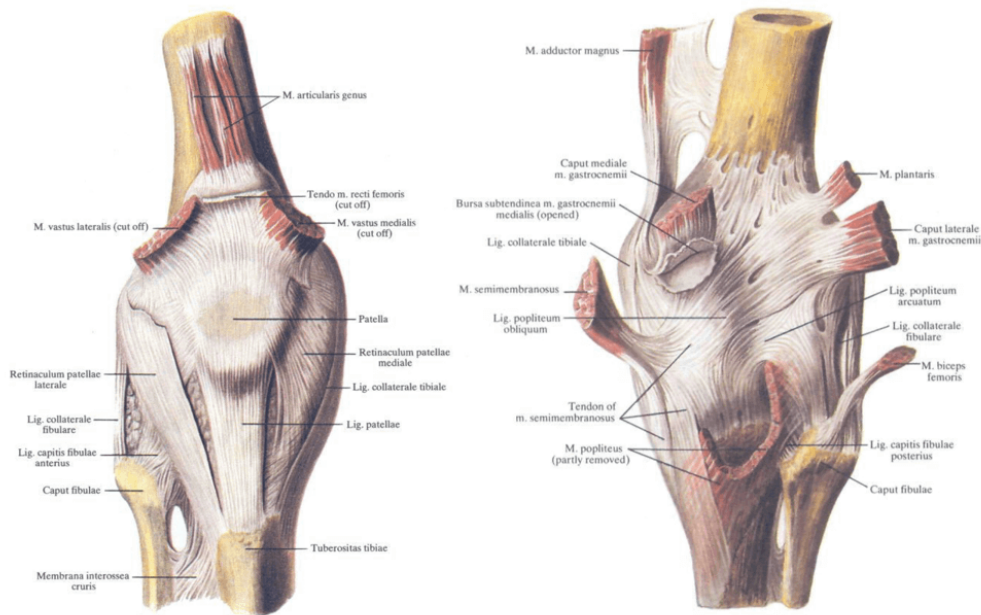


Figure 1.1: Anatomy of the human knee [Sin88]

frequent sports-related knee injuries, may be forced to halt his or her activity for six months or more. During this period, it is highly recommended that patients enroll in a physical therapy program, as a good rehabilitation is absolutely crucial for a successful recovery.

## 1.2 Motivation

It is the responsibility of the physiatrist to guide the physical therapy program, evaluating the patient's condition and adjusting the exercises and procedures accordingly. When it comes to knee rehabilitation, the protocol is quite similar for most injury types, with great focus on restoring the joint's full function, including range of motion (ROM), and regaining muscle mass [PB07]. Joint inflammation following the injury, or even the surgical intervention itself, may cause (and usually does) some limitation of the ROM. And since the patient has to temporarily refrain from exerting load on the injured leg, if not completely immobilize it, muscle atrophy is usually inevitable following a knee injury [KJR<sup>+</sup>07].

Given that not all patients have the same requirements or recover at the same pace, the rehabilitation programs should be as adaptive as possible to their needs. That can only be achieved through frequent assessment and evaluation by the health care professional, mostly the physiatrist. Ideally, there should be a balance between both quantitative and qualitative evaluation. While the first is more detailed and provides more insight on the patient's current status, it is generally harder to perform. Some accurate measurements are time-consuming or can only be performed using expensive specific equipment that not all clinics have available.

Thus, in most cases, the evaluation is mainly qualitative, based on not-so-accurate visual estimations, and also not very frequent. This work is supported by the idea that an easier and quicker way to perform meaningful measurements would contribute to an increased amount of evaluations performed, as well as a deeper insight on the patient's progress.

### 1.3 Goals

This project's main goal is the creation of a mobile application capable of performing accurate measurement of the knee's range of motion. It should be a low-cost solution that is simple and quick enough to operate, so that physical therapists, even more that physiatrists, can use it frequently, as they are the professionals that most closely follow and deal with the patient during the treatment.

Specifically, the app should be capable of estimating the joint's flexion angle using a common smartphone's camera. The estimations should be executed in real-time, without requiring photography or any interaction by the professional other than pointing the camera, and the obtained results should promptly be presented on the device's screen.

### 1.4 Report Structure

This introduction is followed by seven more chapters. The next one consists of a detailed review of state of the art solutions currently aiding the rehabilitation process of knee injuries. Chapter 3 presents a theoretical background relevant to this dissertation. Chapter 4 is the problem statement, where the proposed solution is described in detail, followed by a work plan. Chapters 5 and 6 go through the technical details of the implementation of the solution. And, lastly, chapter 7 concludes with a brief summary and a discussion of possible follow-up work in the future.

## Introduction

## Chapter 2

# State of the Art

---

<b>2.1</b>	<b>Measurements in Knee Rehabilitation . . . . .</b>	<b>5</b>
<b>2.2</b>	<b>Smartphone-based Solutions . . . . .</b>	<b>7</b>
<b>2.3</b>	<b>Conclusion . . . . .</b>	<b>9</b>

---

This chapter is a detailed revision of state-of-the-art tools related to the problem being addressed in this report. It starts with a broad revision of the measurements described in section 1.3 and the techniques or tools used to perform them, followed by a more detailed analysis regarding low-cost alternatives based on smartphones.

### 2.1 Measurements in Knee Rehabilitation

While there is a wide variety of measurements and tests that are relevant to the treatment and recovery of knee injuries, the scope of this project is focused on goniometry — the medical term that refers to the measurement of range of motion of a joint.

#### 2.1.1 Goniometry

The measurement of the range of motion of the knee is a recurrent task in knee rehabilitation which can be achieved using a variety of methods or tools. It is conducted by determining the angle described by two lines representing the axis of the femur and the axis of the fibula, respectively [GBRN87]. The technique generally accepted as reference consists of direct observation of the bones in a radiograph [EGD<sup>+</sup>04], as seen in figure 2.1.

Since knee goniometry should be performed frequently during the rehabilitation, using the radiographic method is not feasible due to radiation exposure. Alternatively, multiple tools were

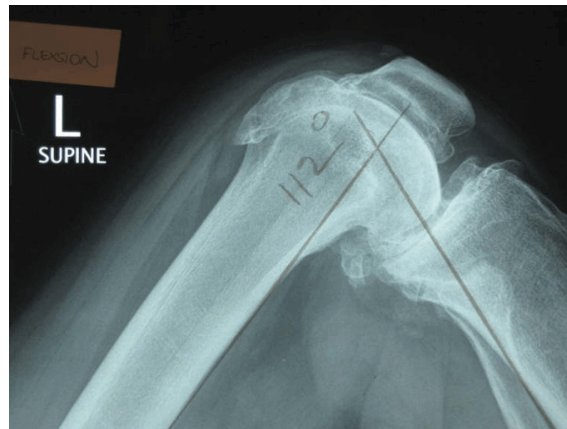


Figure 2.1: Measurement of joint angle in x-ray [NKA<sup>+</sup>11]

created in order to enable this procedure [HDM49]. Currently, the universal goniometer, depicted in figure 2.2, is the one most commonly used. With more than 30 degrees of knee flexion, there is no significant difference between the results obtained using this method and the radiographic one [Enw86].

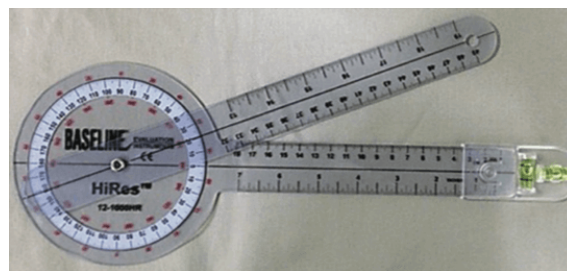


Figure 2.2: Universal goniometer [RSJSL13]

While using this tool is considered one of the most reliable methods, this approach has some disadvantages. First of all, it requires a specific device, which might not be available at every physical therapy clinic. Furthermore, its usage requires manual manipulation by the expert. On the one hand, this introduces some error to the estimation. Thus, it is advisable that all measurements on a patient be performed by the same person [HDM49]. On the other hand, this method can only be used while the patient's injured leg is in a static position. It provides no way to measure the angles of the knee during motion, such as walking or running on the treadmill.

Another device that can be used to measure joint angles is a digital inclinometer, or protractor. This works in the same fashion as a common bubble inclinometer, but reports the actual value of the inclination in degrees. Unlike the universal goniometer, this tool doesn't immediately reveal the angle of the joint. Instead, both the inclination of the thigh and the lower leg must be measured. Then, the desired value can be obtained by subtracting these two. As well as being a bit more complex to use, this device is also more expensive than a universal goniometer. However, in terms of reliability and validity, the two approaches are similar [KH12].

<sup>1</sup>Source: [https://www.msi-viking.com/Mitutoyo-950-318-Pro-360-Digital-Protractor-Output\\_p\\_22350.html](https://www.msi-viking.com/Mitutoyo-950-318-Pro-360-Digital-Protractor-Output_p_22350.html)

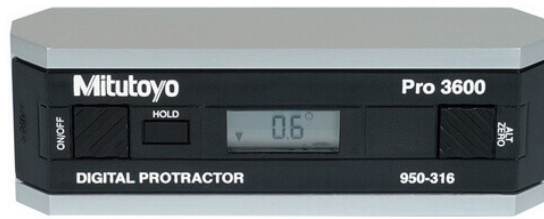


Figure 2.3: Digital protractor<sup>1</sup>

Furthermore, a study published in 1985 concluded that goniometry could be performed with greater accuracy using photography than with standard tools [FW85]. However, it was only a few years later that this approach was practical, thanks to the advances in digital photography. In order to establish the angle of the joint to be measured, two techniques are used. One uses adhesive skin markers that are manually placed on the patient's leg, while the other requires the estimations of the lines of the femur and the fibula, much like in the radiographic approach. Both strategies yield similar consistency and accuracy when compared to using x-rays [NKA<sup>+</sup>11], even considering the estimations done by the expert regarding the angle described by the joint, either when placing the markers or drawing bone lines. One major disadvantage of photography-based goniometry is the necessity of using software to obtain the actual measurement from the captured image. While the results can indeed have great accuracy, errors can easily be introduced by using inappropriate procedures. For instance, the photography should be taken from a perpendicular viewpoint and centered in the angle being measured [DCG06].

## 2.2 Smartphone-based Solutions

Nowadays, information technology (IT) has advanced to a point where mobile communication and portable computation can be combined on a single device known as smartphone. Most of these are packed with a diverse set of sensors, including digital cameras, magnetometers and accelerometers. This makes these devices quite suitable for a huge variety of applications and purposes, including health care. Whether it aids the retrieval of information, supports diagnosis or provides any other kind of solution, the use of smartphones in medicine is growing at a quick pace [MYS12].

The following sections present recent smartphone-based solutions to the problems being approached in this project.

### 2.2.1 Inertial-based goniometry

An inertial sensor that is present in most smartphones is the gyroscope. This component relies on Earth's gravity to determine the orientation, and it is usually calibrated to read a value of  $0^\circ$  when the device is on a horizontal position. Thus, a gyroscope-equipped smartphone can provide the same functionality as a digital inclinometer, making it suitable to perform goniometry. Again, this technique also requires the measurement of two inclinations (thigh and lower leg) before

determining the joint's angle, as depicted in figure 2.4. There are some applications developed specifically for the purpose of goniometry that simplify this process. One of the first apps to be scientifically evaluated is iGoniometer (which is no longer available). The study compared this application with a long arm goniometer<sup>2</sup> and concluded that the difference in accuracy was low enough so that it would not have a negative clinical impact [HSO12].

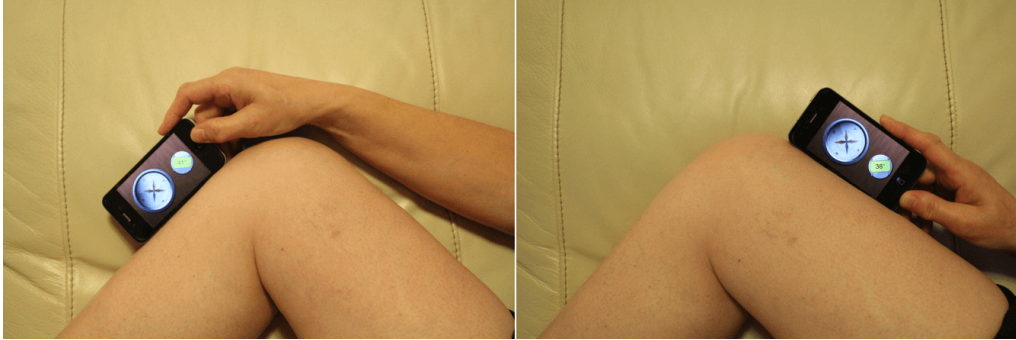


Figure 2.4: Knee goniometry using a gyroscope-based application [Jen13]

A study evaluating a different mobile application, performing knee goniometry at six different leg positions (full extension, 30°, 60°, 90°, 110° and maximal flexion), indicates a slightly higher accuracy of this measurement technique when compared to conventional ones [Jen13]. Previous research demonstrates that this approach is precise enough to be frequently used in a clinical environment. Moreover, there are multiple similar applications available for free, making this an inexpensive solution for performing knee goniometry (excluding the cost of the smartphone).

### 2.2.2 Photography-based goniometry

DrGoniometer is a mobile application that is capable of measuring the angles of the patient's articulations. It uses a photography-based approach, thus being appropriate for static joint angle measurement only. As it was earlier mentioned, in order to get an accurate estimate the camera's point of view should be parallel to the plane defined by the leg [FVS<sup>+</sup>13]. Doing so will avoid errors induced by perspective distortion. It's worth noting that this rule is not specific to DrGoniometer, but rather any photography-based angle measurement system.

In this particular case, after taking a picture, three markers must be dragged to the appropriate positions. Recommended reference points are the center of the knee, the hip and the ankle [FVS<sup>+</sup>13]. Having these three markers positioned, the application presents two dashed lines representing the angle of the joint, and the measured value is visible on the bottom of the screen. Using the coordinates of the markers, the angle calculation is based on simple trigonometry.

Regarding this particular solution, studies have been published demonstrating its reliability and validity, not only for measuring knee range of motion, but also other joints of the human body [OAB<sup>+</sup>15]. The results reveal measurement errors similar to those of traditional measurement tools,

<sup>2</sup>A device similar to the universal goniometer



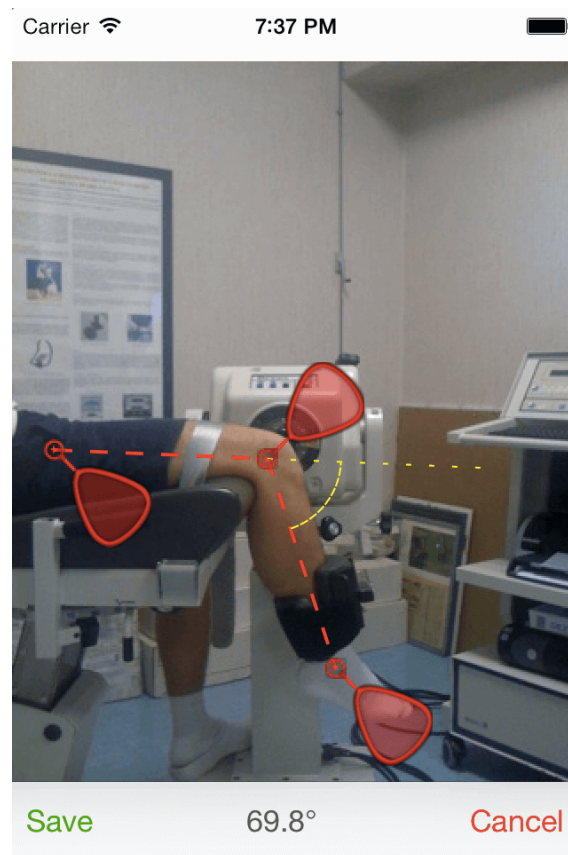


Figure 2.5: Knee ROM measured with DrGoniometer

such as the Universal Goniometer, making it a perfectly valid solution to be used by professionals [FVS<sup>+</sup>13].

## 2.3 Conclusion

This chapter explained one of the most common measurement tasks when treating knee injuries — goniometry — and reviewed existing methods and tools to support it. It started with traditional approaches before moving to more technologically advanced smartphone-based ones, which closely relate to the presented solution. All methods are analyzed in terms of both accuracy and usability, which are the two main factors to considering when choosing one. The next chapter introduces some concepts that are relevant in the context of this dissertation.

## State of the Art

## Chapter 3

# Background

---

3.1	Image Recognition . . . . .	11
3.2	Transfer Learning . . . . .	14
3.3	Deep Learning with Synthetic Data . . . . .	14
3.4	Supporting Frameworks and Software . . . . .	14
3.5	Conclusion . . . . .	16

---

This chapter contains a review of the literature and theoretical concepts that are relevant in the context of this dissertation.

### 3.1 Image Recognition

Computerized image recognition is essential in the context of this dissertation, as the proposed solution consists in using the smartphone’s camera to accurately perform body measurements. The following sections review the theoretical concepts on this topic that are related to the solution being proposed.

#### 3.1.1 Computer vision

Computer Vision is the field of computer science that focuses on the task of enabling computers to process and understand digital images and videos, usually with the ultimate goal of automating human tasks. This involves automatic extraction and analysis of features from the digital data, such as patterns, and a consequent retrieval of relevant information from them. It is a scientific field with innumerable applications across many different areas, such as medicine, robotics, security, biometrics, autonomous vehicles, etc. Computer Vision, however, tries to solve inherently difficult

problems, since the human visual system is excellent in many tasks (e.g. face recognition) and achieving equivalent performance is hard [Hua96]. Among the most typical problems Computer Vision tries to solve are image recognition, motion analysis, scene reconstruction and image restoration. While there are other approaches, many computer vision techniques used nowadays take advantage of machine learning algorithms.

### 3.1.2 Machine learning

Machine learning is a part of the artificial intelligence field which focuses on using techniques that allow the computers to automatically update their behavior in order to improve its performance on a given task, much like the way humans, and other living beings, learn. This is usually based on numerical and statistical algorithms that allow a computer to extract information from data. This is applied in situations where explicitly programming an algorithm to solve a given problem is hard or even impossible. Hence its relevance in the computer vision.

These algorithms can be grouped in two major categories: supervised learning and unsupervised learning. In the first case, by analyzing a given set of labeled data, commonly referred to as the *learning dataset* or *training dataset*, the machine should be able to perform accordingly on new examples it has never seen before. Thus, the learning dataset should be representative and of considerable dimension, which is not always easy to obtain [Lan88]. In the second case, the available data is not labeled and the algorithm itself must find structure in its input.

There are many distinct algorithms, with variable complexities and applications. When it comes to computer vision problems, **deep learning** has recently become one of the most popular and successful branches of machine learning.

### 3.1.3 Deep learning

It was just in recent years that the deep learning field had considerable advances and became a state-of-the-art solution for multiple digital image and video related problems. Thus, it is still a relatively young field of machine learning. Nevertheless, remarkable results have been achieved by resorting to this type of algorithm. It differs from other machine learning algorithms mainly for its use of hierarchical structures, representing data with multiple levels of abstraction [SPP18]. This project aims to explore and develop a solution for the previously described problem using one of the main categories of deep learning techniques — convolutional neural networks (CNN).

### 3.1.4 Convolutional neural networks

CNNs were inspired by the animal visual system's structure, consisting of multiple layers of connected neurons. They have been demonstrated to perform surprisingly well on image-related tasks. Each layer receives input data and passes a transformed output to the next layer, until the end is reached. These structures have achieved impressive results, especially in some computer vision tasks such as face recognition and object classification. A common CNN has three types of neural layers: convolutional layers, pooling layers and fully-connected layers [VDDP18]. In

## Background

convolutional layers, multiple kernels are used to extract feature maps from parts of the image. The purpose of pooling layers is the reduction of the spatial dimensions. The output volume is smaller than the input volume. This is achieved by combining multiple adjacent values into a single one. The strategy used for this process is what differs between different types of pooling layers. The most often used are average pooling and max pooling, as research has shown these have the most positive impact on the CNN's performance.

Convolutional and pooling layers account for most of a CNN's structure and are responsible for extracting and learning relevant features from the images. At the end of the network, there are usually a few fully connected layers. In contrast to the other two kinds of layers, here every neuron of a layer is connected to all the neurons of the next layer. These are responsible for the CNN's reasoning and producing the final output, or *prediction*. One of these layers converts the bidimensional feature map (extracted from the image by the previous layers) to a unidimensional vector which can then be mapped to a set of categories for classification, or serve for further processing. Figure 3.1 shows an example of the structure of a CNN.

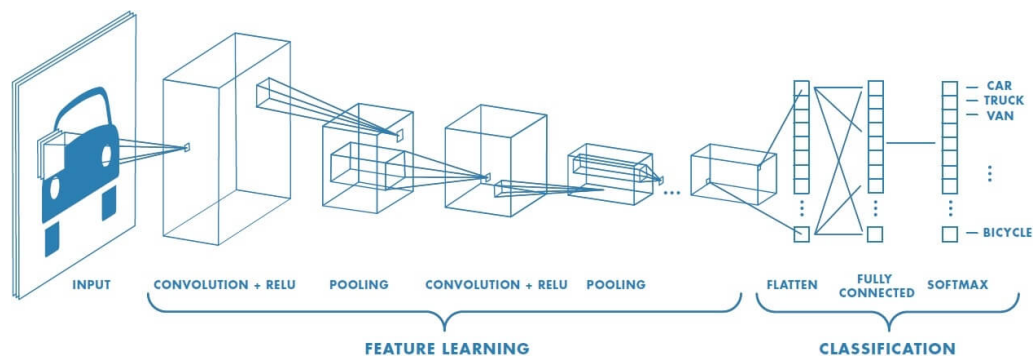


Figure 3.1: Typical structure of a CNN<sup>1</sup>

The way each layer transforms its input to its output is dependent on a number of parameters, commonly called *weights*. As with any machine learning algorithm, CNNs need to be trained with labeled data, and in particularly high amounts. The training of a CNN consists of an iterative process where, after running each image through all the layers, the weights are adjusted in an attempt to match the prediction with the expected outcome (the labels). To evaluate the quality of the predictions, CNNs use a loss function to compare them with the labels. On an untrained model, the initial weights are usually random, so it is expected that the results of the first iterations are not good. However, it is expected that the loss values progressively decrease, describing an exponential decay curve, as the weights become more properly tuned. Eventually, they should converge to a (preferably low) value, meaning that it's no longer improving and the learning process should stop [CWMG18].

<sup>1</sup><https://www.mathworks.com/discovery/convolutional-neural-network.html>

## 3.2 Transfer Learning

The requirement for large amounts of training data is one of the main disadvantages of convolutional neural networks, and the reason why they are not a viable solution in many cases. There is, however, a strategy to help circumvent this issue, called **transfer learning**. This consists of, instead of training models from scratch, starting with a pre-trained model [PY10]. Some popular models, created and trained during machine learning competitions, are publicly available. Most were trained on the ImageNet dataset. Transfer learning not only makes the training process faster and less resource-hungry, it also helps achieving good results with smaller datasets. While training, some layers of the CNN can be frozen, which stops it's weights from being adjusted. Alternatively, they can be left unfrozen and, thus, the weights will be adjusted. This is commonly known as **fine tuning** [SRG<sup>+</sup>16].

## 3.3 Deep Learning with Synthetic Data

Another workaround for the lack of large amounts of data, with which positive results have previously been demonstrated, is synthetic data generation. If using artificially generated images, dataset size should no longer be a problem. This, of course, is not the ideal approach and bring its own set of problems. Specifically, it might be tough to get the CNNs to generalize from what is learned on synthetic data to real data, which is the ultimate goal. The data generation algorithm must produce realistic samples that closely resemble reality in order to make this possible. Ekbatani et. al demonstrated a successful application of this strategy in a problem of counting the number of pedestrians in street photos [KEPS].

## 3.4 Supporting Frameworks and Software

This section provides an overview on some frameworks and software relevant for the solution being proposed.

### 3.4.1 MakeHuman

MakeHuman<sup>2</sup> is a free open-source 3D computing software designed for creating and prototyping realistic humanoid characters. The current version is developed in Python using OpenGL and the Qt framework, which enables it to run on all main operating systems. It has a very simple GUI, where most of the configurable parameters of the 3D models, such as height, weight, muscle volume, etc., are managed through sliders. This program is often used for producing virtual human characters for 3D games or animated motion pictures. It has a plugin-based architecture, but a lot of assets are built-in. Figure 3.2 shows a preview of its main interface.

---

<sup>2</sup><http://www.makehumancommunity.org>

## Background

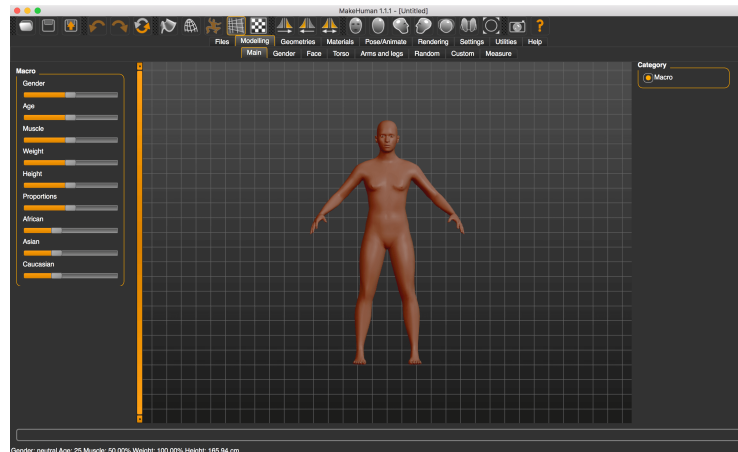


Figure 3.2: MakeHuman’s main interface

Nowadays it is a stand-alone application, but the project started of as a plugin for Blender (which is reviewed next), until it eventually grew too large to maintain with that architecture. During that period, it was awarded the Suzanne Award for the best Blender Python script in 2004.

### 3.4.2 Blender

Blender<sup>3</sup> is another open-source 3D software. However, unlike MakeHuman, which is focused on the creation of 3D characters, Blender provides all the necessary tools to create animated films, visual effects, 3D applications and even video games, since it features an integrated game engine. It’s been one of the state-of-the-art tools in 3D computing for several years. Instead of dealing with a single 3D object or model, in Blender the user is able to control the entire scene, with as many objects as desired, light sources, cameras, etc. Just like MakeHuman, Blender also has a modular architecture. With the appropriate plugin, it is possible to import 3D models created with MakeHuman, in order to pose or animate them.

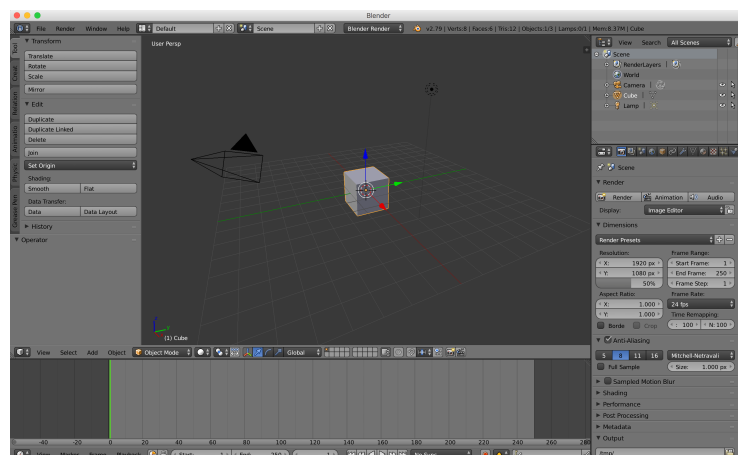


Figure 3.3: Blender’s main interface

<sup>3</sup><https://www.blender.org>

Blender is a professional tool of great complexity, providing an enormous set of features. Hence, it also has a relatively steep learning curve associated with it, as the large amount of panels and options available can be overwhelming for unexperienced users. Figure 3.3 shows the initial interface of Blender's current version.

### 3.4.3 Keras

Keras<sup>4</sup> is a Python open-source library developed for simplifying the use of deep learning. The creators describe it as a high-level neural network API, which runs on top of one of three other deep learning libraries: TensorFlow, CNTK or Theano. Keras makes it very simple and fast to set up and train neural networks, both recurrent and convolutional, and supports running from a GPU, which really boosts performance when it comes to deep learning tasks. Besides providing a simple API for the user to create his own models, it also bundles some popular models, which can be loaded pre-trained, which really facilitates the use of transfer learning.

### 3.4.4 CoreML

CoreML<sup>5</sup> is a machine learning framework introduced by Apple, Inc. in 2017 with the release of the 11th version of the iOS mobile operating system. It allows the integration of previously trained machine learning models on mobile applications. This framework is the basis for other domain-specific frameworks, such as Metal Performance Shaders, which includes support for deep learning and convolutional neural networks, taking advantage of the processing power of the device's integrated GPU.

## 3.5 Conclusion

This chapter briefly presented an approach to perform image recognition, one of the main computer vision problems, based on deep learning. It also reviewed a strategy to circumvent the lack of an abundant dataset, required by CNNs, which consists in the generation of artificial training data using 3D modeling software. Additionally, it reviews available frameworks upon which the new solution is built. The following chapter provides a detailed description of the problem being targeted and summarizes both the proposed solution and the evaluation procedures.

---

<sup>4</sup><https://keras.io>

<sup>5</sup><https://developer.apple.com/documentation/coreml>



## Chapter 4

# Problem Statement

---

<b>4.1</b>	<b>Requirements</b>	<b>17</b>
<b>4.2</b>	<b>Solution Proposal</b>	<b>18</b>
<b>4.3</b>	<b>Evaluation</b>	<b>18</b>
<b>4.4</b>	<b>Conclusion</b>	<b>18</b>

---

This chapter describes the planned approach for developing a low-cost smartphone-based solution to the presented problem. It starts with a summary of the goals and requirements, followed by an overview of the proposed solution, focusing on the main tasks, challenges and possible strategies to overcome them, taking advantage of the concepts and technologies reviewed in the previous chapter. It also presents a comparison with the existing solutions mentioned in chapter 2, explaining the advantages of the one being proposed. Moreover, it describes how the new solution will be evaluated and validated.

### 4.1 Requirements

According to the context and problem described in chapter 1, and as previously stated, the ultimate goal of this dissertation is the creation of a low-cost smartphone-based solution. Specifically, we aim to develop a mobile application that aids physiatrists and physical therapists during the usually long rehabilitation process that follows a knee injury, allowing them to estimate knee range of motion with nothing more than a smartphone.

Using the app should be simple: after selecting the desired type of measurement, the examiner will just have to point the device's camera at the patient, immediately having the results presented on the screen. Unlike the existing photography-based solutions reviewed in chapter 2, this approach

should not require photographs or videos to be captured. Instead, the calculations shall be executed in real-time.

## 4.2 Solution Proposal

Considering the requirements explained above, the first and biggest challenge is performing accurate image recognition. The system must be able to detect when the camera is pointing at a human leg, recognizing the shape and retrieving the desired information, which are the coordinates of three key points in the image (the knee, the center of the thigh and the center of the lower leg) required for the calculation of the joint's angle.

Existing literature describes multiple techniques for performing computerized image recognition. In this dissertation, we decided to explore an approach based on deep learning using convolutional neural networks. The usage of this machine learning technique for image recognition is increasing fast, as research has been revealing surprisingly positive results. Recently, this has become a very common approach in medical image analysis [LKB<sup>+</sup>17]. However, large datasets are required to adequately train convolutional neural networks. Since there is no suitable dataset available, we will synthesize one, taking advantage of 3D modeling software reviewed earlier. As a final step, the solution should be implemented as a smartphone application, taking advantage of Apple's CoreML framework, reviewed in chapter 3.

## 4.3 Evaluation

Since the neural networks will be trained using synthetic data, the main goal is two fold: (i) to achieve high accuracy in synthetic images, and (ii) to accurately transfer/generalize these capabilities for real images. A non-artificial validation set will be created after capturing images of the leg, simulating a treatment environment, and manually labeling them by marking the expected coordinates to be compared with the neural network's predictions. At a more advanced stage, once the proposed solution is implemented and a functional mobile application is developed and ready to be used, it should be tested by actual physical therapists and physiatrists from a few local clinics, with the collaboration of Fábio Rocha, a physical therapist working as an external advisor in this dissertation. During the course of multiple therapy sessions with patients recovering from knee injuries, range of motion shall be measured using both the proposed solution and the traditional methods for comparison.

## 4.4 Conclusion

As explained in the previous sections, this novel solution has some key advantage points when compared to state of the art alternatives, especially in terms of usability. The strategy relies on recent developments in the field of image recognition, where the use of deep learning has been showing

## Problem Statement

impressively positive results. The upcoming chapter is the first one covering the implementation details of this project, starting with synthetic data generation.

## Problem Statement

## Chapter 5

# Synthetic Dataset Generation

---

<b>5.1</b>	<b>3D Model Creation</b>	<b>21</b>
<b>5.2</b>	<b>Manipulation and Posing</b>	<b>22</b>
<b>5.3</b>	<b>Automated Data Generation</b>	<b>23</b>
<b>5.4</b>	<b>Conclusion</b>	<b>25</b>

---

This chapter focuses on the generation of the synthetic image dataset with which the convolutional neural networks were trained. As previously stated, a 3D model of the human body was used in order to generate large amounts of realistic, anatomically accurate and varied image samples, as required by the deep learning technique that was applied.

### 5.1 3D Model Creation

The first step of the data synthesis process was obtaining a 3D model of the human body. This was achieved using the open-source software MakeHuman<sup>1</sup>, a tool developed for the creation of three-dimensional characters. The program initially presents the user with a basic humanoid model, which was then manipulated to more accurately resemble an actual human being, since MakeHuman allows the user to tweak a large set of parameters regarding the shape and volume of body parts, textures, materials and even clothing. Not only does this model have a realistic human body shape, but it is also rigged, meaning the 3D mesh is bound to a digital skeleton consisting of bones and joints, which makes it possible to pose the character according to the requirements. Figure 5.1 shows both the initial and final characters, as well as the corresponding skeleton. This software, however, offers very limited options in terms of posing, only allowing the user to choose

---

<sup>1</sup><http://www.makehumancommunity.org>

from a small set of predefined poses. Since the goal is to synthesize images similar to those a physical therapist would capture with the smartphone in order to measure the knee's flexion angle, it is essential to manipulate the position of the character's legs, especially the knee joints. This was achieved with a different software, as explained in the next section.

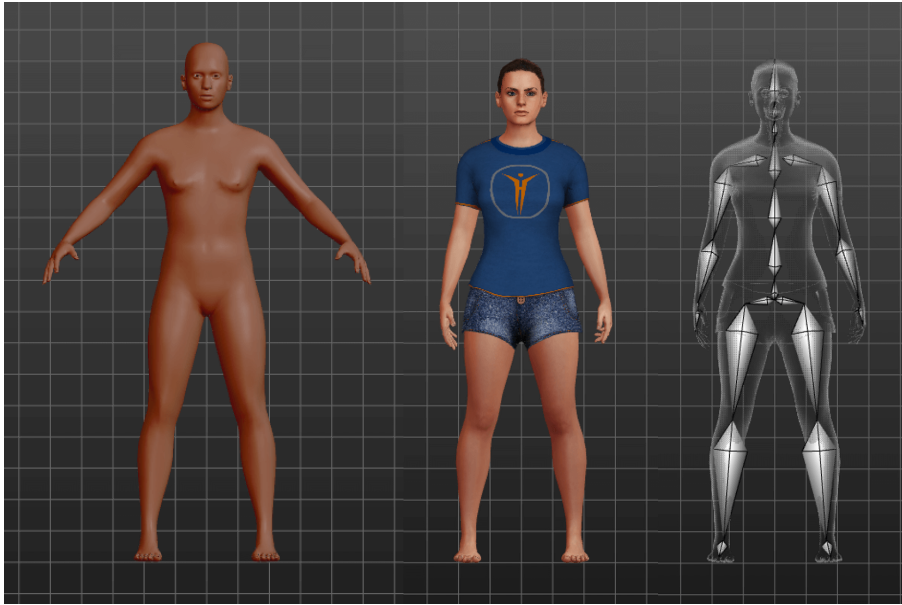


Figure 5.1: MakeHuman 3D models

## 5.2 Manipulation and Posing

In order to manipulate the 3D model created with MakeHuman, it was imported to another open-source software, Blender<sup>2</sup>, via the MHX2 format. This program provides all the necessary features to set up an appropriate scene for generating the image samples. It allows the user to select and manipulate individual bones of the character's skeleton. Different angles of flexion were easily applied to the knee by simply rotating the lower leg bone. However, applying this single transformation generated, in many cases, unrealistic poses. When a patient is lying on their back on a treatment table, flexing the knee also requires a flexion of the hip. Thus, a 1:2 relation between the hip's and the knee's flexion angles was defined, ensuring the heel was permanently aligned with the torso, as if it was resting on the treatment table, as seen in figure 5.2.

At this point, an image resolution of 200x150 pixels was established<sup>3</sup>, and the scene's camera in Blender was configured accordingly. One important concern was making sure that the entire leg was captured by the camera, regardless of the angle of flexion. This was achieved using Constraints, a Blender feature that allows an object's properties to be controlled using variables (such as the properties of a different object). Specifically, an invisible object, consisting of a single point in space, was inserted into the scene to serve as the target of the camera, which is always pointing in

<sup>2</sup><https://www.blender.org/>

<sup>3</sup>This decision is explained in § 6.3 (p. 30)

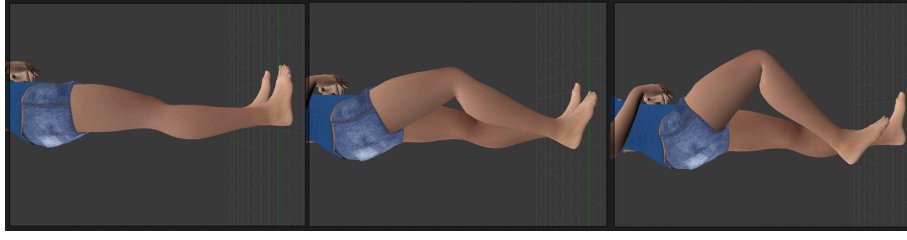


Figure 5.2: Different knee flexion angles in Blender

its direction from a perpendicular perspective, according to its constraints. This point's position is also controlled by constraints: it permanently matches the centroid of the triangle described by the knee, the hip and the ankle. This approach, combined with an appropriate distance of the camera, guarantees that the leg is always centered and fully captured.

Besides the camera and the character, the 3D scene also contains one light source positioned above the model to create appropriate lightning conditions. Additionally, in order to obtain more diverse data, multiple different-toned skin textures included with the MakeHuman software were used. Some examples are presented in figure 5.3.



Figure 5.3: Examples of different skin textures used

### 5.3 Automated Data Generation

Having the scene set up, the dataset generation task was automated by taking advantage of Blender's Python API. A simple script was developed to repeatedly render images from the scene's camera, while randomly applying different flexion angles between each iteration. For every rendered image, the program also writes to a comma-separated values (CSV) file the coordinates of the three points of the leg necessary for determining the flexion angle: **the centers of the thigh, the knee and the lower leg**. Blender's API provides methods to retrieve the bi-dimensional position of an object in the camera's frame. The returned values are normalized, in the interval  $[0,1]$ , and the origin corresponds to the image's bottom-left corner. However, writing to the file, they are converted to absolute pixel values, by multiplying each coordinate by the corresponding image dimension

## Synthetic Dataset Generation

(200x150 pixels), and also transposed to a coordinate system where the origin is the top-left corner of the image, as commonly observed in computer graphics.

$$\begin{aligned}x' &= 200x \\ y' &= 150(1 - y)\end{aligned}\tag{5.1}$$

In order to match each image with its set of coordinates, its filename, which is a numerical index (excluding extension), is also stored in the CSV file. Thus, the data is structured as shown in table 5.1:

Table 5.1: Example of data stored in the CSV file

img	thigh_x	thigh_y	knee_x	knee_y	leg_x	leg_y
0	62	74	98	71	137	73
1	64	69	98	61	132	73
2	85	62	99	30	114	64
3	70	67	98	45	129	67
...	...	...	...	...	...	...

At the top of the Python script, some parameters can be adjusted in order to configure the data generation process. These include the number of samples to be generated, the range of possible knee flexion angles and also a maximum rotation offset, relative to the patients leg, for the camera's position. This offset is used because, in a real-case scenario, it is virtually impossible for the therapist to set the camera in an exact perpendicular direction to the plane defined by the patient's leg. Thus, at each iteration, not only a random flexion angle is computed, but also a rotation for the camera. This rotational range should, however, be small, as significant rotations cause perspective distortion, making it impossible to accurately measure the real angle in the image.

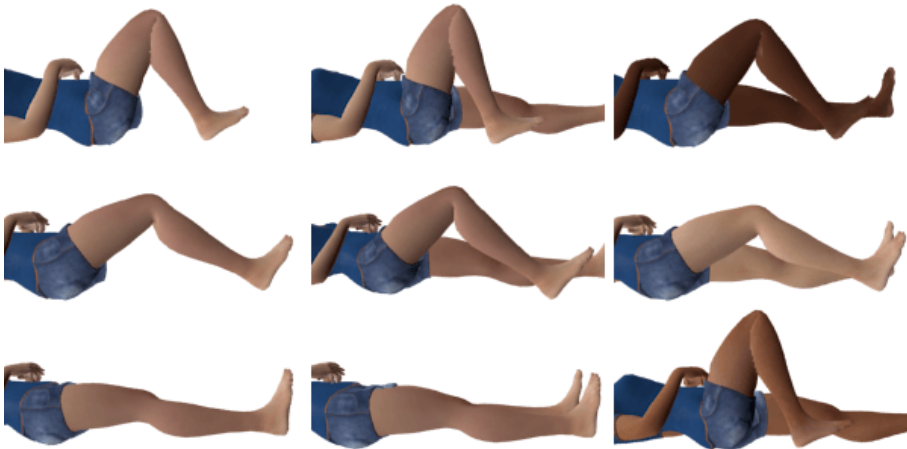


Figure 5.4: Examples of synthesized data



Figure 5.4 shows different types of samples that were obtained using this script: in the first column, only the front leg is visible (an attempt to simplify the deep learning process); in the middle one, both legs are visible. However, while the front leg is posed with random flexion angles, the other leg is straight in a resting position. And in the last column, also both legs are visible, but a different skin texture is randomly chosen for each sample. All images have a fully transparent background and were exported in Portable Network Graphic (PNG) format.

### 5.4 Conclusion

The presented process constitutes an easy method of obtaining large amounts of data for training the neural networks. However, the size of the dataset is not the only relevant aspect. While the images look considerably realistic, they are all very similar: the only variations are the flexion angle, the camera angle and the skin tone. The next chapter explains additional approaches to improve the quality of these datasets in terms of diversity.

## Synthetic Dataset Generation

## Chapter 6

# Image Recognition

---

<b>6.1</b>	<b>Supporting Libraries and Resources . . . . .</b>	<b>27</b>
<b>6.2</b>	<b>Image Augmentation . . . . .</b>	<b>28</b>
<b>6.3</b>	<b>CNN Architectures . . . . .</b>	<b>30</b>
<b>6.4</b>	<b>Training . . . . .</b>	<b>33</b>
<b>6.5</b>	<b>Conclusion . . . . .</b>	<b>43</b>

---

After going through the challenge of obtaining a suitable amount of data for the solution being presented, the following sections review the actual process of attempting to build a convolutional neural network that is capable of recognizing the three relevant coordinates mentioned earlier in real images, which ultimately allows the calculation of the knee's flexion angle.

### 6.1 Supporting Libraries and Resources

This part of the project was implemented using the Python 3.6 programming language, along with some useful frameworks and libraries. The core of this solution is built using Keras 2.2.0, a popular deep learning Python library, running on top of Google's TensorFlow 1.8.0. This stack provides simple and straightforward prototyping of convolutional neural networks, which greatly simplified the process of testing multiple different approaches. While it's API allows the user to create custom CNN models, the framework has available implementations of a few popular architectures, such as MobileNet, DenseNet, VGG16 and InceptionV3, among others. Additionally, it is possible to load these models with weights pre-trained on ImageNet, a massive image dataset commonly used in visual object recognition research. The following sections explain how these features were used in the development of the solution. The management and manipulations of images and data structures

was supported by NumPy and Pillow, two Python libraries widely used in scientific computing. All training tasks were executed on a dedicated nVidia GeForce GTX 1080 Graphics Processing Unit.

## 6.2 Image Augmentation

As stated earlier, synthesizing a dataset has the advantage of easily obtaining a large number of samples. However, introducing enough diversity to enable the neural networks to generalize what is learned from this data to real-life examples is a challenge. Adding to the strategies used during the generation step (flexion angle, camera position and skin texture), some image augmentation techniques were applied during the training process, which is commonly referred to as *online augmentation*. These are:

- Horizontal flipping, with probability of 50%
- Random horizontal translations, with  $\Delta x \in [-20, 20]$
- Random vertical translations, with  $\Delta y \in [-10, 30]$
- Random rotations, with  $\alpha \in [-30^\circ, 30^\circ]$
- Random pictures as background

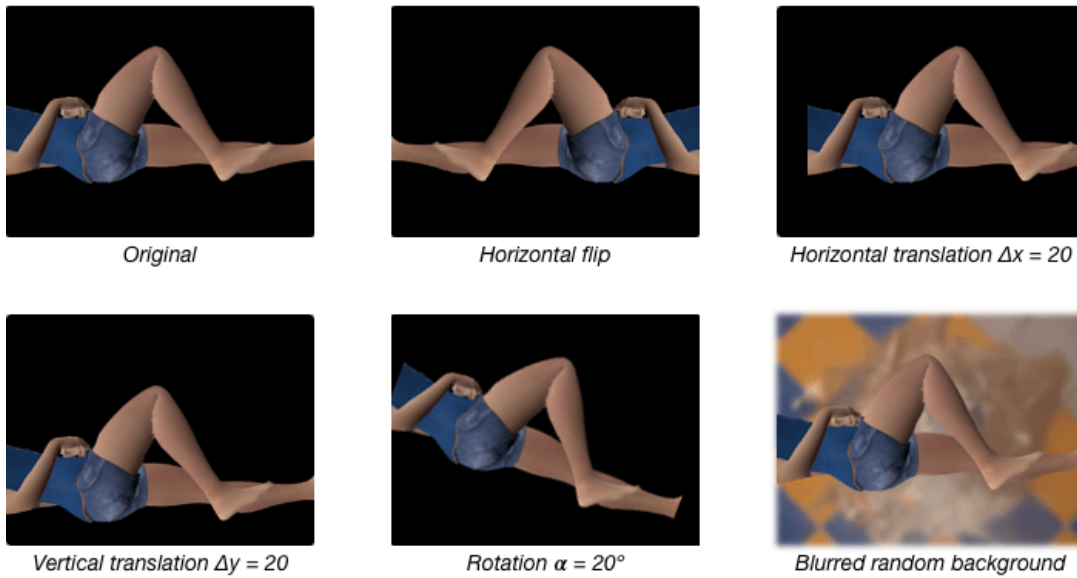


Figure 6.1: Examples of the transformations applied to the dataset

For each image that was loaded from the dataset for training, any of these transformations would be randomly applied and the labels updated accordingly. Multiple tests have been executed, with different combinations of transformations being applied, and also with variable ranges of values for

the translations and rotations. Figure 6.1 shows the results of applying these transformations to a sample of the dataset.

While all techniques aim at improving the quality and diversity of the dataset, each has a specific theoretical goal. While the original dataset only contains images of the right leg of the character, applying horizontal flips should help the network learn how to recognize images of both legs. Rotating should, for instance, improve the results in images where the patient is not lying down on the treatment table or the camera is not properly leveled. Translations should help the network generalize for cases where the leg is not exactly centered in the image.

Horizontal flips are a type transformation that requires special attention, since it changes the order of the labeled points. The original order is first the thigh, then the knee and lastly the lower leg. For the neural networks, however, this order has no meaning and we could easily encounter a situation where the results are terrible, even though the network did perform good predictions... but in the *wrong* order. For instance, if the expected output was  $[(1,0), (0,0), (-1,0)]$  and the network predicted  $[(-1,0), (0,0), (1,0)]$ , the calculated *loss* would be 2, when it should actually be 0, since the three points match perfectly. Thus, it is important that the networks are trained to always predict in the same order. In all other samples, with the rest of the transformations, the labels are ordered from left to right ( $x$  values are in increasing order). This problem has a simple solution: when a horizontal flip is applied, the labels of the first and third points are swapped. This way we ensure the labels are always ordered from left to right, maintaining a pattern which will help during the training process.



Figure 6.2: Some of the photographs used as background

One of the aspects that make all the training samples so similar is the background. All images were generated with a transparent background, having 4 channels: red, green, blue and alpha (RGBA image). However, before they can be used for training, they must be converted to have only 3 channels: red, green and blue (RGB images). The Pillow library provides a method for this conversion, which by default sets the previously transparent pixels to full black. If a network is trained using these images, the performance on real images where the background is not uniform

and black will certainly be poor. The network should be able to recognize which image pixels correspond to the patient's leg and disregard the rest, as they have no meaning. Hence the last transformation mentioned — background change. To circumvent this issue, each time an image is loaded for training, it is pasted over a random photograph before the RGBA to RGB conversion. These photographs were provided by [JDS08], a dataset with 1491 personal holidays photos. Figure 6.2 shows some of these images.

This is the first technique to be applied, so, the background is also affected by all the following transformations. Before merging them with the leg samples, a gaussian blur filter with a 2px radius is applied to reduce the amount of detail of the pictures and making them smoother.

Again, multiple tests were executed in order to assess the impact of these transformations. We started by applying none, and progressively enabling them, one by one, while also experimenting with different combinations. While this should, in theory, improve the results, that's not necessarily the case, as the results will demonstrate.

### 6.3 CNN Architectures

During the execution phase of this project, different CNN architectures were tested and compared, in an attempt to achieve the best results. Given the characteristics of the available dataset, it was decided to start by using transfer learning instead of training a network from scratch. That process was simplified by the Keras framework, which bundles some popular CNN models pretrained with the ImageNet dataset. Two of those were selected: InceptionV3[SVI<sup>+</sup>] and VGG16[SZ14]. A third, custom model, was also used. This is a much simpler model than the other two, and it was trained from scratch on the synthetic dataset. This custom model will be referred to as **Eva**.

While the Eva model was designed for this particular situation, InceptionV3 and VGG16 had to be slightly modified in order to be used — specifically, the first and last layers. The first layer because they were trained on images with a different resolution than those in our dataset, which are 200x150 pixels. This resolution was decided taking into account the minimum size accepted by common CNN models such as the ones being used. For instance, while the minimum size suitable for VGG16 is 48x48 pixels, InceptionV3 requires, at least, 139x139 pixels. The chosen resolution should be large enough to be used with any model, while being, at the same time, small enough not to significantly slow down the training process. Thankfully, Keras is prepared for this situation, and allows the user to specify a custom *input shape* when loading pre-trained models. And also the last layer, since both VGG16 and InceptionV3 were designed for the ImageNet classification problem, which has 1000 distinct classes of images. Thus, both have a final layer with exactly 1000 neurons, which is inappropriate for the problem being solved here. Our problem is different in two aspects: first, instead of predictions on 1000 classes, we need only 6 predictions (the  $x$  and  $y$  coordinates for the centers of the thigh, knee and lower leg); second, this is a regression task, not classification. In order to match these requirements, the last layer must have exactly 6 neurons and no activation function, so the network outputs the untransformed numerical values. Figure 6.3 represents this structure.

## Image Recognition

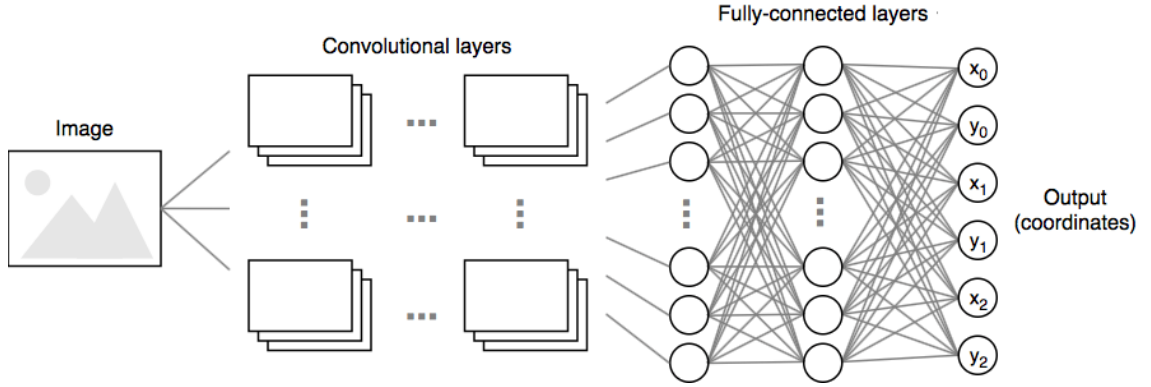


Figure 6.3: Diagram representing the required CNN structure

When a Keras model is created or modified, it must then be compiled before being used. During this step, the user must specify a loss function and an optimizer to run during the training process. With all three models tested, the chosen optimizer was RMSProp[WMH<sup>+</sup>17], which comes bundled in Keras. As for the loss, a custom function was implementing for calculating the **total euclidean distance** between predicted  $(x'_i, y'_i)$  and the expected  $(x_i, y_i)$  three points, according to (6.1).

$$\ell = \sum_{i=0}^2 \sqrt{(x'_i - x_i)^2 + (y'_i - y_i)^2} \quad (6.1)$$

### 6.3.1 InceptionV3

As explained earlier, after loading the InceptionV3 model with pre-trained weights, the final layer was changed in order to fit the current problem. Table 6.1 shows the last fully-connected layers after this change.

Table 6.1: InceptionV3 final fully-connected layers (modified)

Layer Type	Neurons	Parameters
...	...	...
GlobalAveragePooling2D	2048	0
Dense	1024	2098176
Dense	6	6150

Since it was decided to use transfer learning in order to both get improved results and reduce training time, at first, only the last three fully-connected layers were set as trainable. This means only the weights of these layers will be fine-tuned for the synthetic dataset. Thus, here's the summary of the network's parameters:

- Total parameters: 23,907,110
- Trainable parameters: 2,104,326
- Non-trainable parameters: 21,802,784

As shown here, most of InceptionV3's parameters are on the convolutional layers, leaving only a small amount of parameters to be trained (roughly 9%). So, instead of training only the fully connected layers, this time the last 60 layers were set as trainable. This raised the percentage of trainable parameters to nearly 45%:

- Total parameters: 23,907,110
- Trainable parameters: 10,605,510
- Non-trainable parameters: 13,301,600

### 6.3.2 VGG16

A similar process was used for the VGG16 model. Table 6.2 shows the fully-connected layers after it was adapted to the problem:

Table 6.2: VGG16 final fully-connected layers (modified)

Layer Type	Neurons	Parameters
...	...	...
Flatten	12288	0
Dense	4096	50335744
Dense	4096	16781312
Dense	6	24582

The convolutional layers on this model were equally frozen. However, a significant difference from InceptionV3 is evident when it comes to the trainable parameters:

- Total parameters: 81,856,326
- Trainable parameters: 67,141,638
- Non-trainable parameters: 14,714,688

Unlike the previous model, the majority of VGG16's parameters are on the fully-connected layers. In this model, about 82% of the parameters are trainable. However, when using Transfer Learning, it is advisable to set all fully-connected layers as trainable, since they act as the classifier or, in this case, as the regressor, and are quite specific to the data domain.

### 6.3.3 Eva

Eva was the name given to the model that was created from scratch. It is much smaller and simpler in comparison to InceptionV3 and VGG16, having only 3 convolutional layers and a total of 11. Code block 6.1 shows the Python script responsible for the creation of this model, using built-in Keras' methods:



## Image Recognition

```
1 Eva = Sequential([
2     Conv2D( filters=32, kernel_size=(3,3), activation="relu", input_shape=(150,200,3) ),
3     MaxPooling2D( pool_size=(2,2), strides=(2,2) ),
4     Conv2D( filters=64, kernel_size=(4,4), activation="relu" ),
5     MaxPooling2D( pool_size=(2,2) ),
6     Conv2D( filters=4, kernel_size=(3,3), activation="relu" ),
7     MaxPooling2D( pool_size=(2,2) ),
8     Flatten( ),
9     Dropout( rate=0.2 ),
10    Dense( units=100, activation="relu" ),
11    Dense( units=50, activation="relu" ),
12    Dense( units=6 )
13 ])
```

Code Block 6.1: The script for generating the Eva model

As can be seen on table 6.3, this produces a simpler and less deep model, with a much smaller set of parameters.

Table 6.3: Layers of the Eva CNN model

Layer Type	Output Shape	Parameters
Conv2D	(None, 148, 198, 32)	896
MaxPooling2D	(None, 74, 99, 32)	0
Conv2D	(None, 71, 96, 64)	32832
MaxPooling2D	(None, 35, 48, 64)	0
Conv2D	(None, 33, 46, 4)	2308
MaxPooling2D	(None, 16, 23, 4)	0
Flatten	(None, 1472)	0
Dropout	(None, 1472)	0
Dense	(None, 100)	147300
Dense	(None, 50)	5050
Dense	(None, 6)	306

However, this model is not pre-trained on any dataset. Thus, all layers must be trained, since weights are randomly initialized. Even so, given the total number of parameters, training this model will be quite less time-consuming than the other two.

- Total params: 188,692
- Trainable params: 188,692
- Non-trainable params: 0

## 6.4 Training

In this phase of the project, the three CNN models will be tested with different combinations of image augmentation techniques applied to the dataset. The training set has 3000 samples and is the same in all tests (only differs by the transformations applied on the fly). The validation set, however, will not be exactly the same in each experiment. It will always be composed of a total

of 500 samples: 25 synthetic images with the original skin texture and only one leg visible; 425 synthetic images with varied skin textures and both legs visible; and 50 real pictures (originally only 3, to which similar image augmentation was applied in advance). The only difference between the validation sets of each experiment is, again, the augmentation techniques applied. For each different combination in an experiment, a matching validation set, with the same transformations applied is used. Figure 6.4 shows some samples from the validation set.



Figure 6.4: Samples from the validation sets

In each of the experiments, the learning curves of the loss values (both on training and validation sets) are presented. The Eva model is represented by the **orange** line, InceptionV3 by the **blue** and VGG16 by the **dark red** one. Along with these curves, a few sample predictions executed by these models are shown, providing a more visual outcome. Lastly, the minimum loss values reached by each model are also documented.

### 6.4.1 Test #1: No augmentation

The first test consisted of training on the original dataset, without performing any image augmentation procedures.

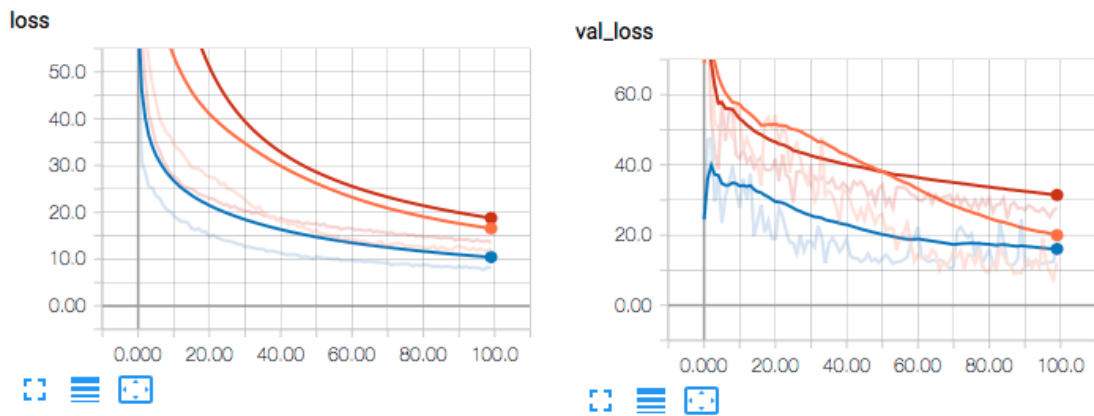


Figure 6.5: Training curves for Test #1. The left and right charts correspond to the training and validation sets, respectively.

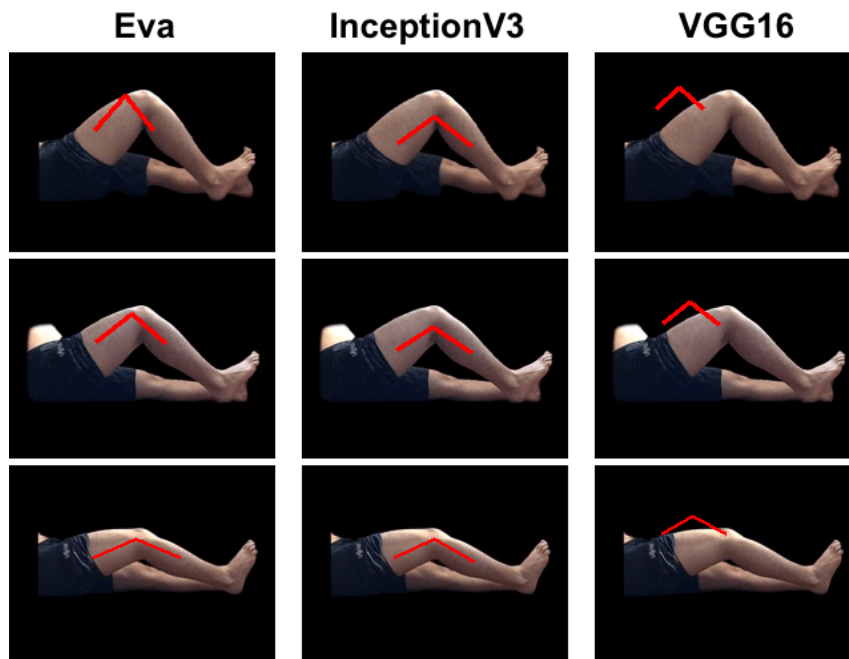


Figure 6.6: Sample predictions after Test #1

The Eva model reached a minimum loss of 12.24 in the testing set and 7.318 in the validation set. The Inception V3 model reached a minimum loss of 8.023 in the testing set and 12.37 in the validation set. The VGG16 model reached a minimum loss of 13.60 in the testing set and 26.94 in the validation set.

### 6.4.2 Test #2: Horizontal flips

In the second test, dataset images are horizontally flipped with a probability of 50%.



Figure 6.7: Training curves for Test #2. The left and right charts correspond to the training and validation sets, respectively.

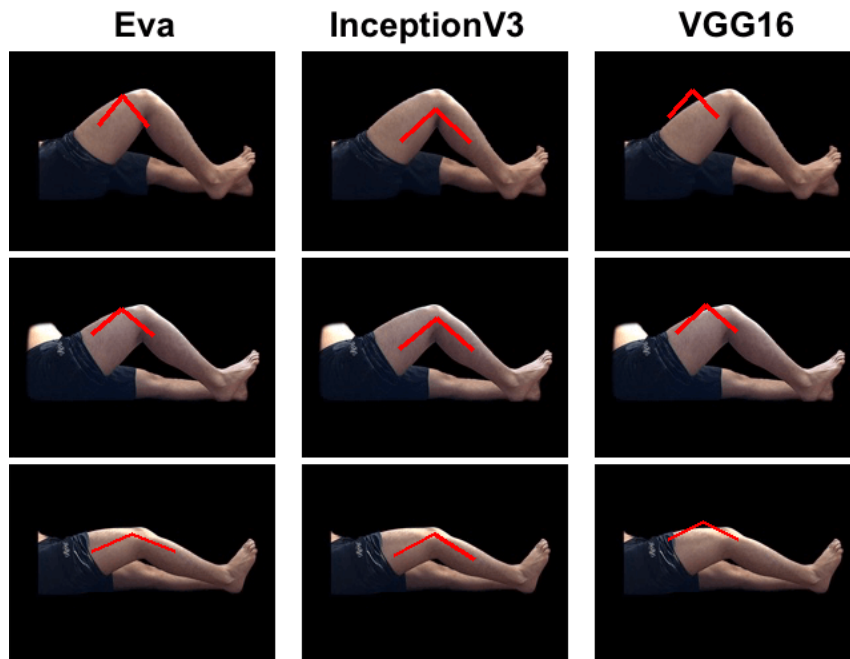


Figure 6.8: Sample predictions after Test #2

The Eva model reached a minimum loss of 13.61 in the testing set and 11.05 in the validation set. The Inception V3 model reached a minimum loss of 7.790 in the testing set and 18.49 in the validation set. The VGG16 model reached a minimum loss of 13.90 in the testing set and 21.60 in the validation set.

### 6.4.3 Test #3: Translations

On the third test, dataset images are horizontally translated a random value of the range  $[-20,20]$  and vertically translated a random value of the range  $[-10,30]$ .

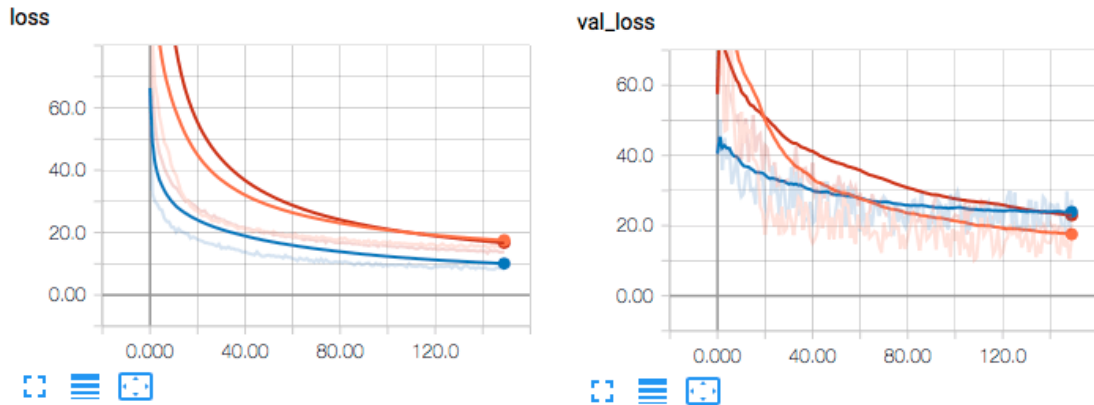


Figure 6.9: Training curves for Test #3. The left and right charts correspond to the training and validation sets, respectively.

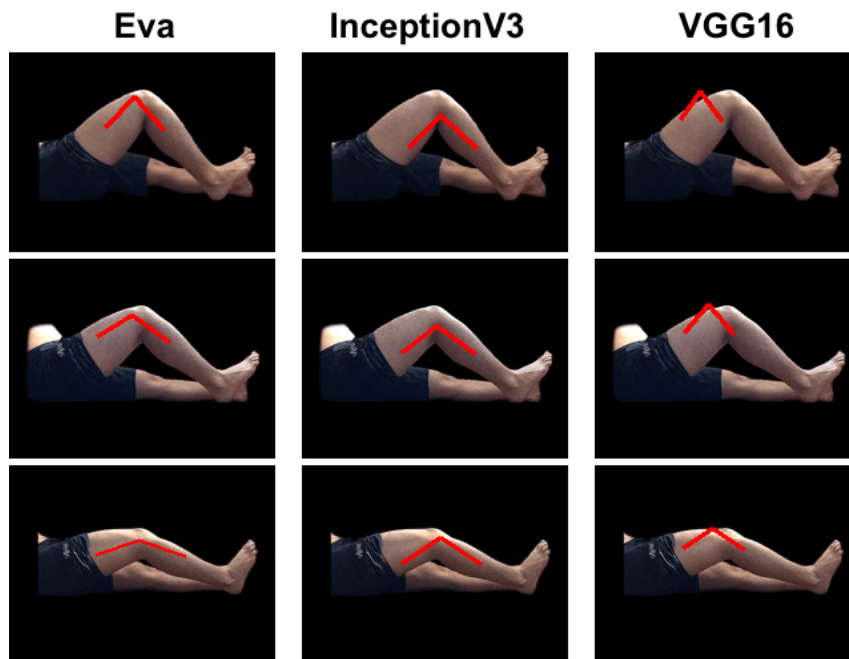


Figure 6.10: Sample predictions after Test #3

The Eva model reached a minimum loss of 15.25 in the testing set and 10.54 in the validation set. The Inception V3 model reached a minimum loss of 8.236 in the testing set and 19.57 in the validation set. The VGG16 model reached a minimum loss of 13.90 in the testing set and 18.70 in the validation set.

#### 6.4.4 Test #4: Background images

On the fourth test, a random background selected from the previously mentioned dataset is added to each training sample.

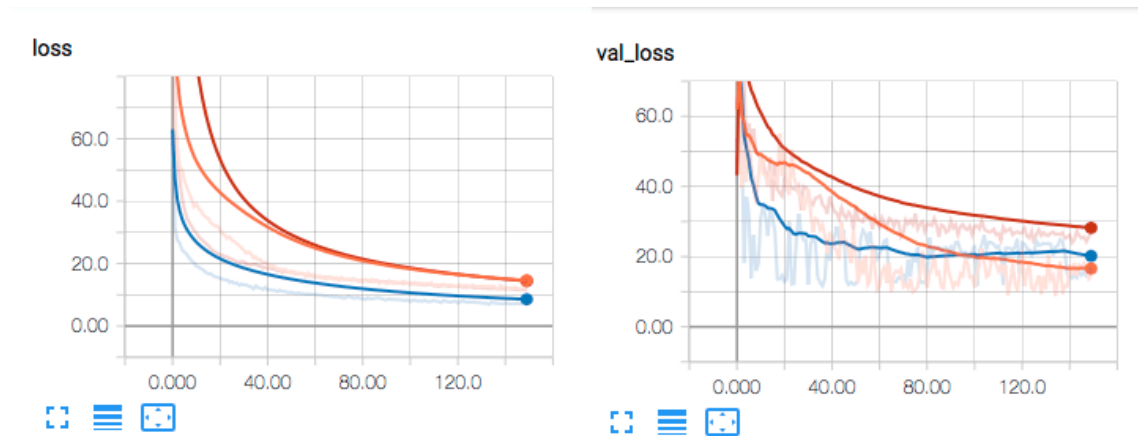


Figure 6.11: Training curves for Test #4. The left and right charts correspond to the training and validation sets, respectively.

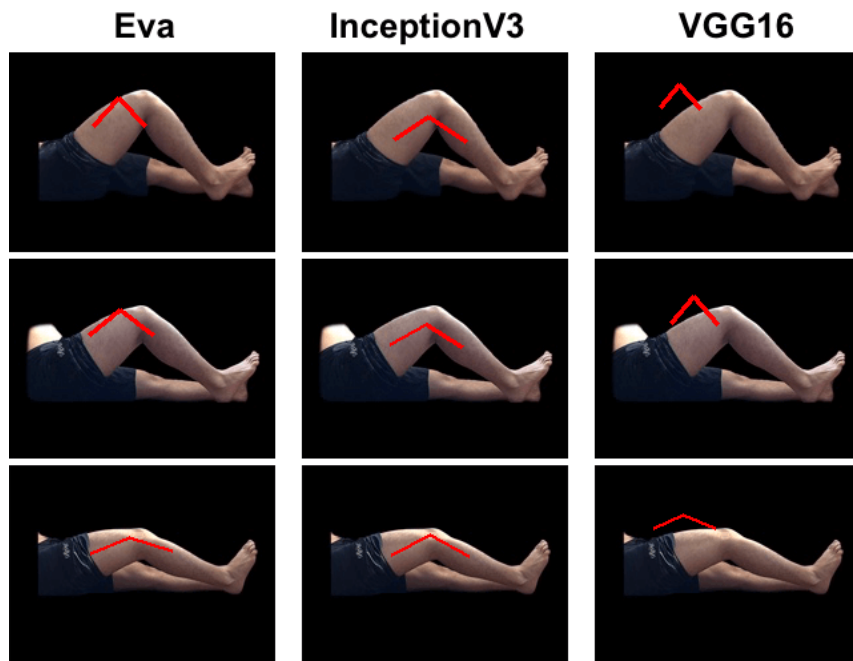


Figure 6.12: Sample predictions after Test #4

The Eva model reached a minimum loss of 11.97 in the testing set and 13.67 in the validation set. The Inception V3 model reached a minimum loss of 6.983 in the testing set and 14.36 in the validation set. The VGG16 model reached a minimum loss of 11.43 in the testing set and 25.69 in the validation set.

### 6.4.5 Test #5: Rotations

On the fifth test, rotations are applied to the dataset samples, relative to their center, of random amplitude in the range  $[-15^\circ, 15^\circ]$

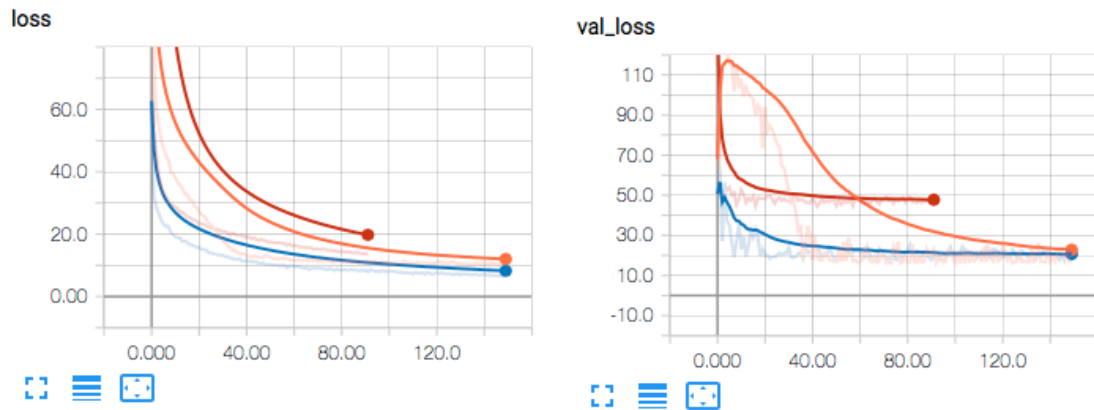


Figure 6.13: Training curves for Test #5. The left and right charts correspond to the training and validation sets, respectively.

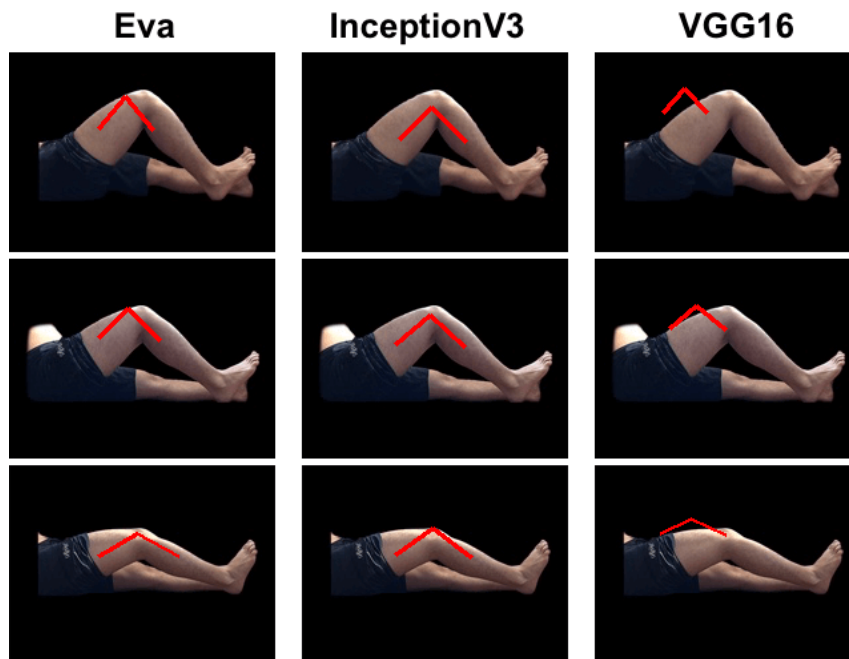


Figure 6.14: Sample predictions after Test #5

The Eva model reached a minimum loss of 10.50 in the testing set and 17.15 in the validation set. The Inception V3 model reached a minimum loss of 6.609 in the testing set and 21.37 in the validation set. The VGG16 model reached a minimum loss of 13.69 in the testing set and 47.42 in the validation set.

### 6.4.6 Test #6: Flips and translations

On the sixth test, both horizontal flips and translations are applied.

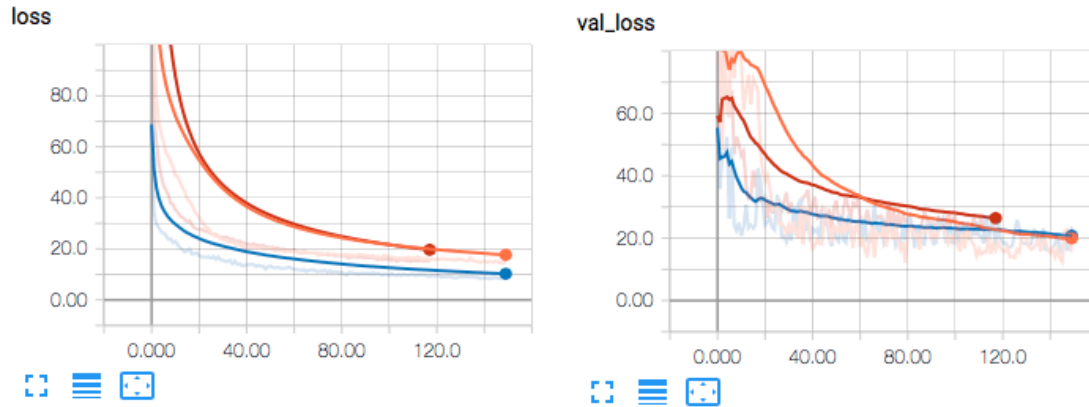


Figure 6.15: Training curves for Test #6. The left and right charts correspond to the training and validation sets, respectively.

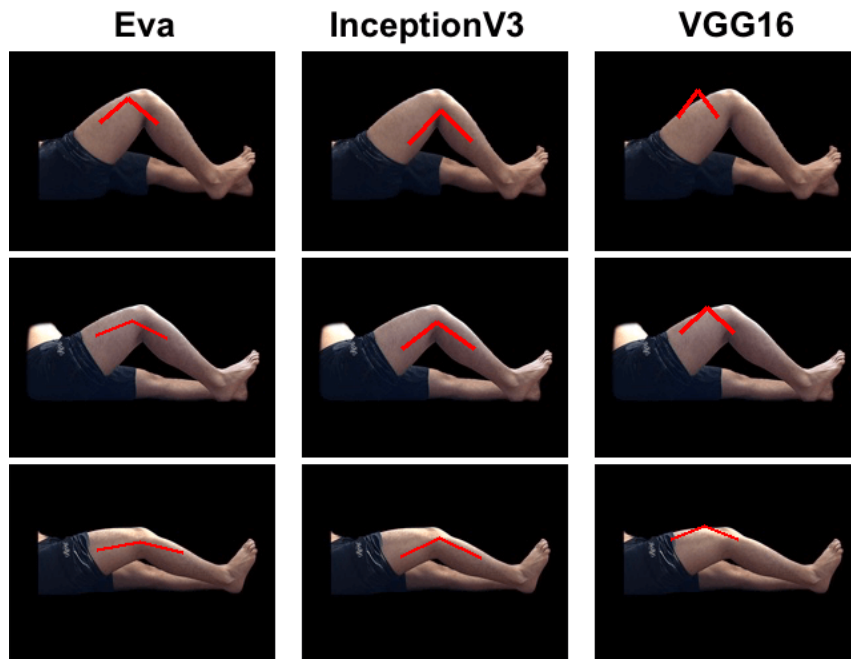


Figure 6.16: Sample predictions after Test #6

The Eva model reached a minimum loss of 15.08 in the testing set and 12.36 in the validation set. The Inception V3 model reached a minimum loss of 8.314 in the testing set and 15.24 in the validation set. The VGG16 model reached a minimum loss of 15.23 in the testing set and 21.94 in the validation set.



### 6.4.7 Test #7: Flips, translations and rotations

On the seventh test, samples get randomly flipped, translated and also rotated.

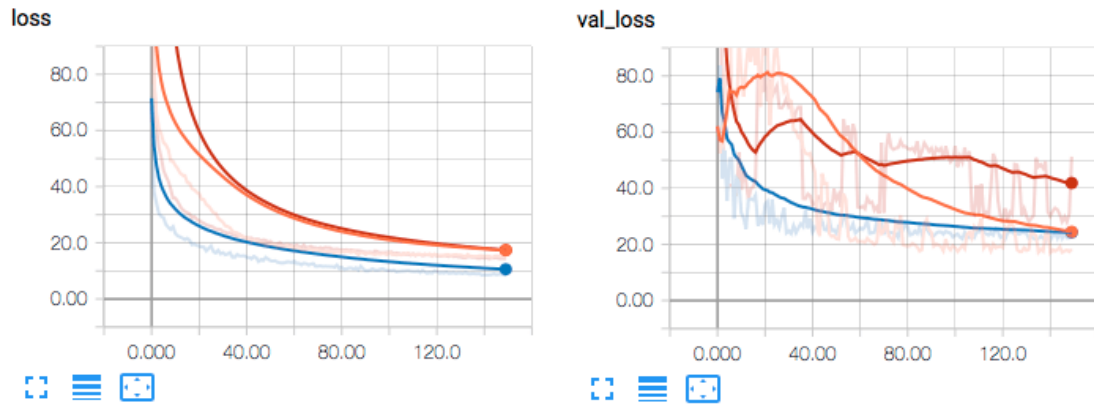


Figure 6.17: Training curves for Test #7. The left and right charts correspond to the training and validation sets, respectively.

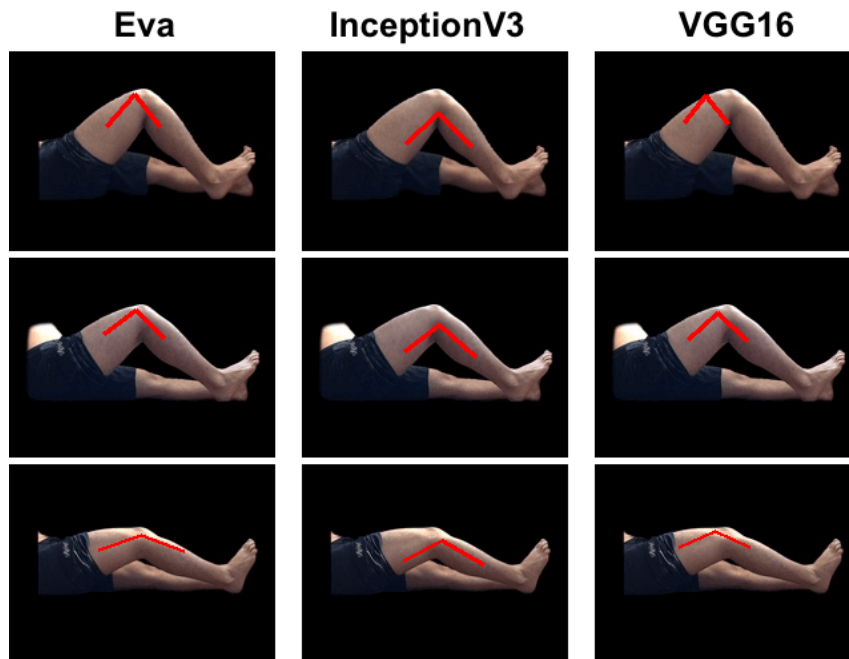


Figure 6.18: Sample predictions after Test #7

The Eva model reached a minimum loss of 14.98 in the testing set and 17.16 in the validation set. The Inception V3 model reached a minimum loss of 8.384 in the testing set and 22.10 in the validation set. The VGG16 model reached a minimum loss of 14.07 in the testing set and 28.12 in the validation set.

### 6.4.8 Test #8: Flips, translations, rotations and backgrounds

Lastly, on the eighth test, all four image augmentation techniques are used simultaneously.

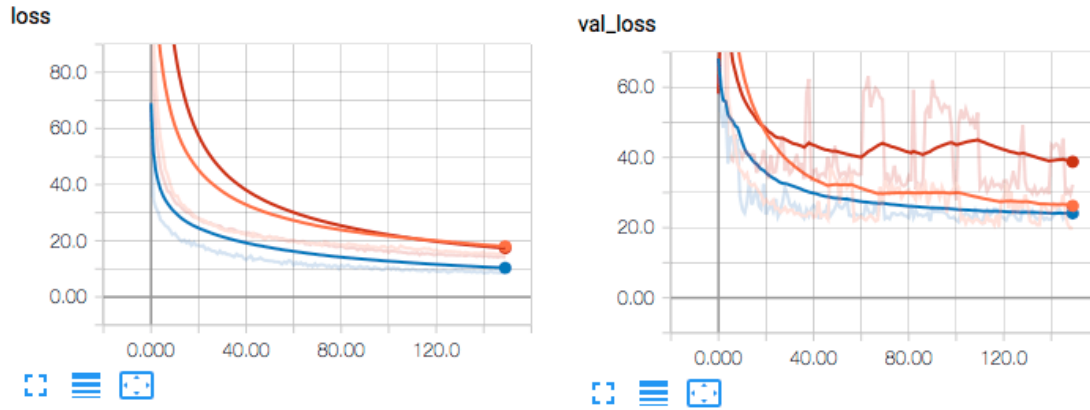


Figure 6.19: Training curves for Test #8. The left and right charts correspond to the training and validation sets, respectively.

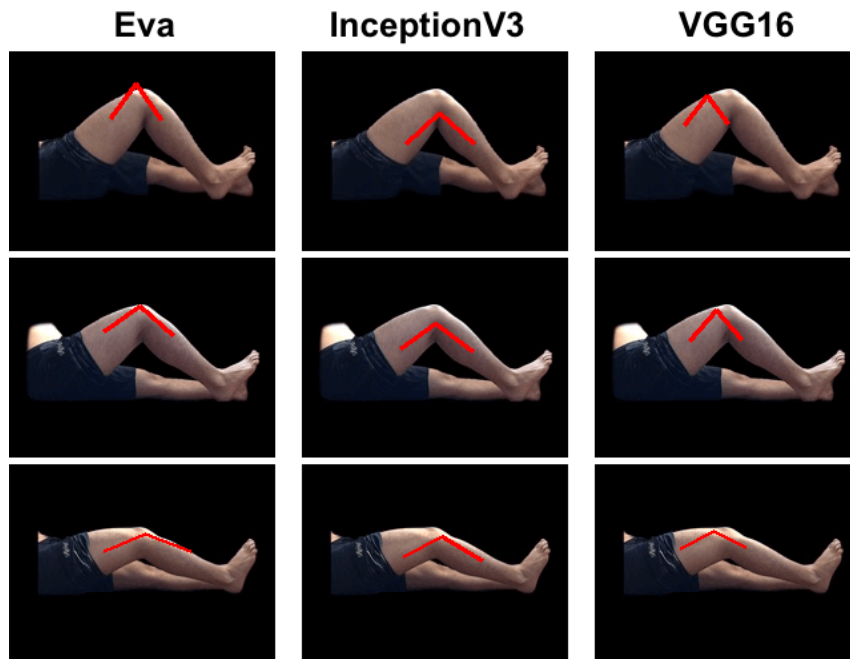


Figure 6.20: Sample predictions after Test #8

The Eva model reached a minimum loss of 15.53 in the testing set and 19.89 in the validation set. The Inception V3 model reached a minimum loss of 8.320 in the testing set and 22.64 in the validation set. The VGG16 model reached a minimum loss of 13.97 in the testing set and 32.22 in the validation set.

## 6.5 Conclusion

As we can observe in table 6.4, regardless of the image augmentation techniques used, InceptionV3 always achieves the best loss values in the training set, while Eva achieves the best results in the validation set. Hence, although the smoothed training lines presented in the previous section might cause a different impression, we can conclude that Eva seems to generalize better to unseen images. However, both these models achieved similarly good results. On the other hand, VGG16 results fell short to the expectations. In some test cases, the training curve did not even converge properly and the predictions hit significantly off target, evidencing difficulty in generalizing to real data.

Table 6.4: Results of the 8 test scenarios. T and V columns represent the minimum loss values on the training and validation sets, respectively. Lower results are better.

Test	I3 T	I3 V	VGG T	VGG V	Eva T	Eva V
#1	<b>8.02</b>	12.37	13.60	26.94	12.24	<b>7.32</b>
#2	<b>7.79</b>	18.49	13.90	21.60	13.61	<b>11.05</b>
#3	<b>8.24</b>	19.57	13.90	18.70	15.25	<b>10.54</b>
#4	<b>6.98</b>	14.36	11.43	25.69	11.97	<b>13.67</b>
#5	<b>6.61</b>	21.37	13.69	47.42	10.50	<b>17.15</b>
#6	<b>8.31</b>	15.24	15.23	21.94	15.08	<b>12.36</b>
#7	<b>8.38</b>	22.10	14.07	28.12	14.98	<b>17.16</b>
#8	<b>8.32</b>	22.64	13.97	32.22	15.53	<b>19.89</b>



Figure 6.21: InceptionV3 predictions on edited photographs (background removed)



Figure 6.22: InceptionV3 predictions on real unedited photographs

## Image Recognition

While there is certainly much room for further improvement, specifically in terms of the diversity of the synthetically generated dataset, this outcome indicates that some good progress has been made. Figures 6.21 and 6.22 are evidence that, while the background on real photographs does interfere with the network's behavior, it seems to work around that fairly well.

While there might be many reasons for that, the fact that InceptionV3, which was pre-trained on ImageNet, achieved a good performance offers some significant indication of the positive impact transfer learning can have. The final results are also aligned with previous research indicating synthetic data generation as a valid alternative for training convolutional neural networks when an appropriate dataset is unavailable.

## Chapter 7

# Conclusions

---

<b>7.1 Difficulties</b>	<b>46</b>
<b>7.2 Future Work</b>	<b>46</b>
<b>7.3 Epilogue</b>	<b>48</b>

---

In this dissertation, we started by familiarizing with the problem of performing accurate measurements in the context of knee rehabilitation. While state of the art solutions are able to provide quite high levels of accuracy, we identified a problem related to poor usability, which has a negative impact on the frequency of this kind of assessment by the physiatrists and physical therapists. Thus, a novel solution was hypothesised, taking advantage of deep learning algorithms for performing computerized image recognition, which would enable the development of a mobile solution requiring a single tool — a smartphone, which is nowadays readily available on virtually every physical therapy clinic. The health professional would simply need to point the camera to the patient's leg in order to obtain an estimate of the flexion angle. Another core aspect of this dissertation is the challenge of acquiring a large dataset which is required by these deep learning algorithms, and the strategy that was used to tackle it — synthetic data generation. Resorting to 3D modeling software, a method for artificially producing a training dataset was developed. Regarding the image recognition task, we had two main goals: achieving high accuracy on the training data and also being able to generalize to real data. With this in mind, we studied and tested three different CNN architectures: VGG16, InceptionV3 and a custom one. In order to enrich the synthesized data, we used a few image augmentation strategies: (i) horizontal flipping, (ii) random horizontal translations, (iii) random vertical translations, (iv) random rotations and (v) random background changes. One key aspect to note is the fact that, during the tests, the actual flexion angle was never measured. Even though it can be calculated using the predicted coordinates, the error of this prediction might not perfectly correlate to the error of the flexion angle. In the end, the results

indicated Eva as the best performing CNN model, with InceptionV3 being very close. Both goals were achieved with relative success. All source code developed for this dissertation is publicly available on a GitHub repository: <https://github.com/joamnb/eva-masters>.

### 7.1 Difficulties

The key challenges encountered in developing this dissertation relate to the difficulty of synthesizing an appropriate dataset. Artificial generation is definitely a valid alternative when real data is scarce, but it is not ideal. Image recognition is a very complex problem on its own, and getting neural networks to learn to deal with real data without actually observing it during training is not easy. The generated data must look as realistic as possible, and in this particular case where we were dealing with a part of the human body, there is really an overwhelmingly large number of factors and variables to consider. However, the final outcome did not fall short of our expectations and the results can be considered quite positive.

### 7.2 Future Work

Despite the satisfactory results obtained, this project is certainly not complete. The initial goal of creating a usable mobile application for physiatrists and physical therapists was not achieved, and the solution should still be refined before taking that step. The following sections provide an overview on possible steps that can be taken in the future in order to improve and validate the solution.

#### 7.2.1 Diversity of the dataset

There is definitely room for improvement when it comes to the dataset that was used. Additional variations should be introduced, specifically using whole different 3D human-body models, with different anatomies. The length of the legs, their thickness, the lack or presence of hair and even the clothing are some of the many factors that could be explored.

#### 7.2.2 Different CNN architectures

The solution can also be improved in terms of the image recognition. One promising strategy would be to try using additional CNN architectures. For each model, there are multiple variables related to the training process that can be tweaked and explored, such as the learning rate or the number of layers that are set as trainable, when using transfer learning. Another aspect to consider is the size of the models. Given that the final goal is the creation of a smartphone-based solution, performance is a concern. Large models such as VGG16 and InceptionV3 might not be suitable to run on such hardware. Smaller architectures, such as MobileNet [HZC<sup>+</sup>17] and ResNet [HZRS16], might be more appropriate for this purpose. Another approach worth exploring consists of performing image segmentation [Zel16], using specific CNNs for this purpose such as LinkNet [CC17]. These could

be used to detect the precise boundaries of the leg, which would enable additional methods for estimating the knee's flexion angle.

### **7.2.3 Predicting vectors instead of points**

While the three points that the CNN architectures were trained to predict enable the calculation of the flexion angle, there are other alternatives worth exploring. Specifically, instead of three points, training for predicting the just coordinates of the knee, as well as two vectors that describe the directions of the thigh and the lower leg. This data is sufficient to calculate the angle as well, however it has a different meaning. Unlike the coordinates, these vectors would, for instance, remain unchanged whenever translations were applied. This could potentially simplify the network's task, improving the outcome.

### **7.2.4 Predicting the flexion angle directly**

Following the same line of thought of the previous hypothesis, instead of three key points, the CNNs could be trained to directly predict the knee's flexion angle. Again, this would equally satisfy the ultimate goal (and even eliminate the need for additional calculations). However, in this case the architecture would have to change, as instead of 6 neurons in the final layer, there would only be 1.

### **7.2.5 Mobile application**

Ultimately, the developed solution should be ported to the mobile platform. With mobile machine learning libraries, such as CoreML, trained CNN models can be integrated in a mobile application to perform estimations using the smartphone's hardware. Ideally, this process should be fast enough that it can be done in real-time. Instead of capturing pictures that would posteriorly be evaluated, the health professional should only be required to correctly aim the camera at the patient's leg in order to get flexion angle estimations on the screen.

### **7.2.6 On-site validation**

Once the mobile application is developed, the solution must be validated in a professional environment. This could be done in several physical therapy clinics, where the therapists would perform measurements with the developed application as well as using traditional methods. The results would be collected and analyzed, in order to determine the real accuracy and reliability of this solution.

### **7.2.7 Augmented reality**

In recent years, there have been great developments in the field of augmented reality. This is a technology worth exploring in this context, as it might improve the current solution. As previously mentioned, one of the main weaknesses of image-based measurements is the error introduced by perspective distortion. However, if the system was able to detect the plane defined by the patient's

## Conclusions

leg and its orientation relatively to the camera, this error could be mathematically compensated. This would no longer require the camera to be positioned in an exactly perpendicular direction, making the solution more versatile and easy to use.

### 7.2.8 Additional measurements

While this solution is focused on goniometry, in the future it could be adapted for additional measurements that are relevant for physical therapists and physiatrists when treating knee injuries. For instance, targeting the problem of muscle atrophy, a similar solution could be explored in order to estimate the relative muscle volume of the patient's legs. Given a frontal picture where both the healthy and the injured legs are visible, the system should be able to estimate the volume ratio between the two, which could, by approximation, provide insight on the muscular balance.

## 7.3 Epilogue

This dissertation had quite ambitious goals set from the beginning, and evidence of that fact is the large amount of strategies and alternative approaches described above that can still be explored for further development and improvement of this solution. In terms of **synthetic data generation** and **image recognition**, which were the two crucial parts of this dissertation, the results were overall satisfactory. Initial expectations regarding the final part of the project, namely the deployment of the solution to a fully functional mobile application fell short, as that phase was not reached. While some interesting results were achieved, my perception is that there is a great margin for enhancement. There is still a long to go, but significant advances have certainly been made in the right direction.



# References

- [CC17] Abhishek Chaurasia and Eugenio Culurciello. LinkNet: Exploiting Encoder Representations for Efficient Semantic Segmentation. *arXiv.org*, pages arXiv:1707.03718–4, June 2017.
- [CWMG18] Weipeng Cao, Xizhao Wang, Zhong Ming, and Jinzhu Gao. A review on neural networks with random weights. *Neurocomputing*, 275:278–287, January 2018.
- [DCG06] Colin Dunlevy, Marese Cooney, and John Gormely. Procedural considerations for photographic-based joint angle measurements. *Physiotherapy Research International*, 10(4):190–200, 2006.
- [EGD<sup>+</sup>04] John Z Edwards, Kenneth A Greene, Robert S Davis, Mark W Kovacik, Donald A Noe, and Michael J Askew. Measuring flexion in knee arthroplasty patients. *The Journal of Arthroplasty*, 19(3):369–372, April 2004.
- [Enw86] C S Enwemeka. Radiographic verification of knee goniometry. *Scandinavian journal of rehabilitation medicine*, 18(2):47–49, 1986.
- [FVS<sup>+</sup>13] Giorgio Ferriero, Stefano Vercelli, Francesco Sartorio, Susana Muñoz Lasa, Elena Ilieva, Elisa Brigatti, Carolina Ruella, and Calogero Foti. Reliability of a smartphone-based goniometer for knee joint goniometry. *International journal of rehabilitation research. Internationale Zeitschrift für Rehabilitationsforschung. Revue internationale de recherches de readaptation*, 36(2):146–151, June 2013.
- [FW85] D R Fish and L Wingate. Sources of goniometric error at the elbow. *Physical Therapy*, 65(11):1666–1670, November 1985.
- [GBRN87] Prem P Gogia, James H Braatz, Steven J Rose, and Barbara J Norton. Reliability and Validity of Goniometric Measurements at the Knee. *Physical Therapy*, 67(2):192–195, February 1987.
- [GR03] John P Goldblatt and John C Richmond. Anatomy and biomechanics of the knee. *Operative Techniques in Sports Medicine*, 11(3):172–186, July 2003.
- [HDM49] F A Hellebrandt, Ellen Neall Duvall, and Margaret Lee Moore. The Measurement of Joint Motion: Part III—Reliability of Goniometry\*. *Physical Therapy*, 29(7):302–307, July 1949.
- [HSO12] K Hambly, R Sibley, and M Ockendon. Level of agreement between a novel smart-phone application and a long arm goniometer for the assessment of maximum active knee flexion by an . . . . *International Journal of Physiotherapy and Rehabilitation*, 2, July 2012.

## REFERENCES

- [Hua96] T Huang. Computer Vision : Evolution And Promise. Technical report, 1996.
- [HZC<sup>+</sup>17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv.org*, page arXiv:1704.04861, April 2017.
- [HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, 2016.
- [JDS08] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Hamming Embedding and Weak Geometric Consistency for Large Scale Image Search. In *Computer Vision – ECCV 2008*, pages 304–317. Springer, Berlin, Heidelberg, Berlin, Heidelberg, October 2008.
- [Jen13] Jean-Yves Jenny. Measurement of the Knee Flexion Angle With a Smartphone-Application is Precise and Accurate. *The Journal of Arthroplasty*, 28(5):784–787, May 2013.
- [KEPS] Hadi Keivan Ekbatani, Oriol Pujol, and Santi Segui. Synthetic Data Generation for Deep Learning in Counting Pedestrians. In *6th International Conference on Pattern Recognition Applications and Methods*, pages 318–323. SCITEPRESS - Science and Technology Publications.
- [KH12] Morey J Kolber and William J Hanney. The reliability and concurrent validity of shoulder mobility measurements using a digital inclinometer and goniometer: a technical report. *International journal of sports physical therapy*, 7(3):306–313, June 2012.
- [KJR<sup>+</sup>07] R Kannus, L Jõzsa, R Renström, M Järvtöen, M Kvist, M Lento, P Oja, and I Vuori. The effects of training, immobilization and remobilization on musculoskeletal tissue. *Scandinavian Journal of Medicine & Science in Sports*, 2(3):100–118, January 2007.
- [Lan88] Pat Langley. Machine learning as an experimental science. *Machine Learning*, 3(1):5–8, 1988.
- [LKB<sup>+</sup>17] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A W M van der Laak, Bram van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, December 2017.
- [MSK06] M Majewski, Habelt Susanne, and Steinbrück Klaus. Epidemiology of athletic knee injuries: A 10-year study. *The Knee*, 13(3):184–188, June 2006.
- [MYS12] Abu Saleh Mohammad Mosa, Illhoi Yoo, and Lincoln Sheets. A systematic review of healthcare applications for smartphones. *BMC medical informatics and decision making*, 12(1):67, July 2012.
- [NKA<sup>+</sup>11] Justine M Naylor, Victoria Ko, Sam Adie, Clive Gaskin, Richard Walker, Ian A Harris, and Rajat Mittal. Validity and reliability of using photography for measuring knee range of motion: a methodological study. *BMC musculoskeletal disorders*, 12(1):77, April 2011.

## REFERENCES

- [OAB<sup>+</sup>15] Simon J Otter, Brunilda Agalliu, Nicola Baer, Georgie Hales, Katrina Harvey, Keeley James, Richard Keating, Warren McConnell, Rachel Nelson, Saddaf Qureshi, Steven Ryan, Abigail St John, Heather Waddington, Katie Warren, and Duane Wong. The reliability of a smartphone goniometer application compared with a traditional goniometer for measuring first metatarsophalangeal joint dorsiflexion. *Journal of foot and ankle research*, 8(1):30, 2015.
- [PB07] K Pfeifer and W Banzer. Motor performance in different dynamic tests in knee rehabilitation. *Scandinavian Journal of Medicine & Science in Sports*, 9(1):19–27, January 2007.
- [PY10] Sinno Jialin Pan and Qiang Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, August 2010.
- [RSJSL13] Sean Roach, Jun G San Juan, David N Suprak, and Marc Lyda. Concurrent validity of digital inclinometer and universal goniometer in assessing passive hip mobility in healthy subjects. *International journal of sports physical therapy*, 8(5):680–688, October 2013.
- [Sin88] R D Sinelnikov. *Atlas of Human Anatomy In Three Volumes - Vol. 1: The Science of Bones, Joints, Ligaments and Muscles*. Atlas of Human Anatomy In Three Volumes. Mir Publishers, 1988.
- [SPP18] Rajat Kumar Sinha, Ruchi Pandey, and Rohan Pattnaik. Deep Learning For Computer Vision Tasks: A review. April 2018.
- [SRG<sup>+</sup>16] Hoo-Chang Shin, Holger R Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M Summers. Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. *IEEE Transactions on Medical Imaging*, 35(5):1285–1298, February 2016.
- [SVI<sup>+</sup>] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826. IEEE.
- [SZ14] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, abs/1409.1556, 2014.
- [VDDP18] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep Learning for Computer Vision: A Brief Review. *Computational intelligence and neuroscience*, 2018:7068349, 2018.
- [WMH<sup>+</sup>17] Olga Wichrowska, Niru Maheswaranathan, Matthew W Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Nando de Freitas, and Jascha Sohl-Dickstein. Learned Optimizers that Scale and Generalize. *arXiv.org*, March 2017.
- [Zel16] A Zelener. Object Localization, Segmentation, Classification, and Pose Estimation in 3D Images using Deep Learning. 2016.